2003

Learning to Exploit Dynamics for Robot Motor Coordination

Michael T. Rosenstein
University of Massachusetts - Amherst

LEARNING TO EXPLOIT DYNAMICS FOR ROBOT MOTOR COORDINATION

A Dissertation Presented

by

MICHAEL T. ROSENSTEIN

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2003

Department of Computer Science

LEARNING TO EXPLOIT DYNAMICS FOR ROBOT MOTOR COORDINATION

A Dissertation Presented

by

MICHAEL T. ROSENSTEIN

Approved as to style and content by:

Andrew G. Barto, Chair

Neil E. Berthier, Member

Andrew H. Fagg, Member

Roderic A. Grupen, Member

Richard E. A. Van Emmerik, Member

W. Bruce Croft, Department Chair
Department of Computer Science

To my parents.

ACKNOWLEDGMENTS

I am grateful for the support and encouragement of many people, especially my thesis advisor, Andy Barto. I enjoy every lunch meeting and impromptu conversation, occasionally on topics far astray from the latest technical challenge of my research. I always learn something new from our time together, and Andy is a wonderful role model of how to study a problem from many different points of view. To seek out or even to simply recognize the connections among diverse scientific disciplines is a valuable skill to instill in a young scientist. Thank you, Andy.

I also thank my thesis committee for their years of inspiration and assistance: Neil Berthier, for giving me a developmental perspective of motor coordination and for many helpful comments and suggestions; Andy Fagg, for support in all aspects of research, from hardware woes to research vision to detailed comments of the manuscript to candid advice about career choices; Rod Grupen, for his energy and enthusiasm about my work and for pointing me in the direction of dynamic manipulability one late-night plane trip from Houston; and Richard Van Emmerik, for our years of teaching together with many lively discussions and for making me aware of the breadth and depth of Bernstein's work on motor coordination.

I am fortunate to have been part of both the Autonomous Learning Laboratory and the Laboratory for Perceptual Robotics, with many talented members as resources for ideas and debate. I am especially grateful for many helpful discussions with Amy McGovern, Mohammad Ghavamzadeh, Ted Perkins, Rob Platt, Balaraman Ravindran, and Khashayar Rohanimanesh.

And of course this whole enterprise would not have been possible without the love and support of my extended family and my friends outside the UMass community. Thank you

all for understanding the crazy life of a graduate student. Without my children, Hannah and Joshua, I could have easily forgotten how to live in the moment and how to enjoy the small things in life. And without Rebecca, I never would have made it over the first hill in this marathon. All my love to you.

This research was supported by the National Science Foundation under Grant No. IRI-9720345 and by a NASA fellowship from Johnson Space Center under Grant No. NAG-9-1379. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation or the National Aeronautics and Space Administration.

ABSTRACT

LEARNING TO EXPLOIT DYNAMICS FOR ROBOT MOTOR COORDINATION

MAY 2003

MICHAEL T. ROSENSTEIN

B.S., BOSTON UNIVERSITY

M.S., BOSTON UNIVERSITY

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Andrew G. Barto

Humans exploit dynamics—gravity, inertia, joint coupling, elasticity, and so on—as a regular part of skillful, coordinated movements. Such movements comprise everyday activities, like reaching and walking, as well as highly practiced maneuvers as used in athletics and the performing arts. Robots, especially industrial manipulators, instead use control schemes that ordinarily cancel the complex, nonlinear dynamics that humans use to their advantage. Alternative schemes from the machine learning and intelligent control communities offer a number of potential benefits, such as improved efficiency, online skill acquisition, and tracking of nonstationary environments. However, the success of such methods depends a great deal on structure in the form of simplifying assumptions, prior knowledge, solution constraints and other heuristics that bias learning.

My premise for this research is that crude kinematic information can supply the initial knowledge needed for learning complex robot motor skills—especially skills that exploit dynamics as humans do. This information is readily available from various sources such as a coach or human instructor, from theoretical analysis of a robot mechanism, or from conventional techniques for manipulator control. In this dissertation I investigate how each type of kinematic information can facilitate the learning of efficient "dynamic" skills.

This research is multidisciplinary with contributions along several dimensions. With regard to biological motor control, I demonstrate that *motor synergies*, i.e, functional units that exploit dynamics, evolve when trial-and-error learning is applied to a particular model of motor skill acquisition. To analyze the effects of velocity on dynamic skills and motor learning, I derive an extension to the notion of *dynamic manipulability* that roboticists use to quantify a robot's capabilities before specification of a task. And along the machine learning dimension, I develop a *supervised actor-critic architecture* for learning a standard of correctness from a conventional controller while improving upon it through trial-and-error learning. Examples with both simulated and real manipulators demonstrate the benefits that this research holds for the development of skillful, coordinated robots.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Humans frequently exhibit remarkable proficiency at complex motor tasks such as handwriting, juggling, and machine assembly. One way to characterize our success with such tasks is that we have a mastery of redundant degrees of freedom (DOF). Even basic motor skills such as grasping an object require the coordination of many DOF, from more than a dozen joint motions to hundreds of thousands of muscle fibers, depending on the level of interest. Mother Nature may supply a means for direct coordination at some levels—by grouping muscle fibers into motor units and by recruiting motor units in a principled way during muscle contraction—but at other levels, an individual's capacity for learning plays a crucial role. We not only learn to cope with excess DOF, we also learn to exploit them for efficient solutions.

Humans acquire new motor skills with relative ease, whereas even simple motor tasks, such as walking and throwing, pose challenges to roboticists and artificial intelligence researchers developing similar skills for robots. When manipulating an object, for instance, a robotic arm must not only deliver the object to the right place, but must often achieve the proper orientation while managing contact forces and obstacles. All of this must be done in the face of uncertainty—uncertainty about the state of the environment as well as uncertainty about the effects of the robot's actions on that environment. The typical solution of path planning and trajectory tracking works well for solving highly structured problems but not for situations with complex, nonlinear dynamics, with inadequate models, and with little prior knowledge of a favorable solution.

1.1 Research Summary

In the broadest sense, the goal of this research is to develop coordinated machines, where "coordinated" refers to efficient motor skills comparable to those acquired by humans. In particular, my goal is to develop the robot analogues of motor "synergies" that exploit, rather than cancel, the natural dynamics of the musculoskeletal system.

Although a great deal of research exists on biological motor control and the effects of training, too little is known at this time to provide a clear prescription of how to build coordinated machines. This leaves us, the robot designers, in a position to select from many available machine learning and adaptive control techniques and to use general principles of motor learning to guide our design strategies. For this work, I demonstrate the use of iterative optimal control methods while assuming as little as possible with regard to the availability of a detailed model of the robot dynamics. More specifically, simple proportional-derivative controllers serve as approximate control knowledge for subsequent optimization by *reinforcement learning* (RL) methods.

For robot manipulators—the class of robots considered in this thesis—the intrinsic state and action spaces are both multi-dimensional and real-valued, the intersegmental dynamics are complicated and nonlinear, and the kinematic goals are non-unique due to the redundant degrees of freedom. Such characteristics pose a problem for general learning methods like RL, although the trade-off is the possibility of finding an otherwise unavailable, yet worthwhile solution. The suitability of RL for a specific, practical application depends a great deal on structure such as simplifying assumptions, prior knowledge, and solution constraints. One goal of my work, then, is to provide the proper structure (in the robot and the learning algorithm) to make a given motor learning problem tractable without closing off advantageous solutions.

The contributions of this research can be characterized along several dimensions. Foremost is the use of *practical* learning techniques for controlling difficult systems such as redundant manipulators. By tackling hard problems with real robots, this work also helps

delineate the kind of problems handled successfully by a class of algorithms. In other words, I demonstrate that certain reinforcement-based methods are suitable for controlling robotic arms. A further benefit of this line of research is that we gain some insights about human motor learning. By manipulating the internal workings of a robot's control system, we develop hypotheses about biological control systems along with ways to test those hypotheses.

In more pragmatic terms, this research offers a framework for acquiring the basic motor skills of a robot manipulator. One area of application is the use of robots in support of space exploration. The improvement of a robot's autonomous capabilities not only enables missions where human intervention is infeasible, but also shortens training time and reduces fatigue for telerobot operators. More specifically, this research is concerned with tasks beyond the capabilities of technologies like the current Space Shuttle remote manipulator system (RMS). The Space Shuttle RMS is a remote-controlled robot that operates at relatively slow speeds, i.e., the movement is largely kinematic and the robot's controller compensates for the manipulator dynamics. However, a central part of this thesis is the exploitation of dynamics, which is useful not only for the purpose of optimizing a performance criterion but also for the purpose of simplifying control inputs from a human supervisor.

1.2 Bernstein's Problem

The Russian physiologist Nikolai Bernstein (1896-1966) contributed a great deal to our present understanding of human movement. His experiments and theories were comprehensive, covering such a wide range of issues as cybernetics, motor programs, hierarchical control, self-organization, dynamics, and learning. Bernstein was dogmatic only in his rejection of Pavlovian conditioning as a basis for control and cognition [26, 57]; he embraced and advanced many competing theories and brought them together under the umbrella of *coordination*.

In Bernstein's words, motor coordination is the "organizational control" of the nervous system over the motor apparatus [21]. An understanding of coordination is not simply a matter of discovering the mechanisms of control. One must also comprehend the combination of mechanisms—the "style" of control [66]—used by the nervous system. But due to the large number of degrees of freedom (joints and muscles) this organizational control is sometimes "complex and delicate" [21]. Thus, the *degrees of freedom problem*, now commonly called Bernstein's problem in the human movement literature, refers to the control of a complex, redundant system as exemplified by the human motor apparatus.

How the nervous system solves Bernstein's problem for highly skilled movements, such as those needed by athletes and musicians, as well as for everyday acts, such as reaching and walking, remains a major thrust of present research on human motor control. At the forefront of this research effort are proponents of ecological approaches and dynamical systems theories (cf. Turvey's "Round 2" of work on Bernstein's problem [146]). Such efforts emphasize relatively simple physical principles—with low-dimensional dynamics—and self-organization as a means to coordination, despite Bernstein's focus on coordination as a "process" or "problem" to be solved by an active participant: "Motor functions comprise a basic group of processes by which the organism not merely reacts to its environment but even acts on the latter.... Each significant act is a solution (or an attempt at one) of a specific problem of action. But the problem of action, in other words the effect which the organism is striving to achieve, is ... the reflection or model of future requirements (somehow coded in the brain)...." [21, p. 171]. Thus, *the task for the nervous system is the design of a motor problem that, once solved through practice or learning, also solves the degrees of freedom problem*.

Bernstein also offered speculation about the role of trial-and-error learning as a general solution to the "motor problem." In particular, he viewed coordination as a process of active search rather than purely random, "passive statistical computation of successes and failures" [21, p. 161]. He recognized that early attempts at a new task may be completely

random, not because these attempts represent a naive effort to find an optimal solution, but rather because they inform the learning system and direct the subsequent search towards the optimum. Bernstein concluded that this sort of trial-and-error learning alone is insufficient as an explanation of coordination. He rejected *tabula rasa* approaches in favor of theories where learning is biased and constrained by structure derived from an individual's experience as well as from innate mechanisms [21, p. 175].

More to the point of this thesis, Bernstein [21, p. 128] also recognized the connection between human coordination and control of machines:

> "If motor co-ordination is a system of mechanisms ensuring the *control* of the motor apparatus and permitting its rich and complex flexibility to be utilized to the full, what can we say at the present time about the means and mechanisms of this *control* of motor acts? How may the regularities we now observe [in higher animals] be employed in the interests of applied cybernetics ... so that we may more precisely illuminate the gap which still qualitatively divides ... automata from ... highly developed organisms?"

Despite the relative simplicity of robot manipulators, with very few "muscles" to deal with, the degrees of freedom problem persists as the basic issue in the design of robot controllers that exploit dynamics as humans do. This thesis, therefore, is focused on questions about what mechanisms form the basis of coordination and what principles or "regularities" from human coordination can be used by robots. And, like Bernstein, the foundation of my technical approach is the formation of a structured learning problem.

1.3 Overview of Technical Approach

Control of robot manipulators typically involves two separate problems: trajectory *planning* and trajectory *tracking*. The planning problem often deals with a kinematic description of movement, whereas the tracking problem deals with manipulator dynamics

without reference to any specific trajectory. As described in Chapter 2, several advantages follow from this decomposition, although "coordination" requires the integration of both kinematics and dynamics. Thus, the approach taken in this thesis combines both planning and tracking by learning a control policy where movement trajectories are implicit in the coupled dynamics of the robot, the task, and the controller.

However, the learning framework does not do away with explicit trajectories altogether. As discussed in Chapter 3, I assume the availability of an initial kinematic path that solves a relaxed version of the movement problem. In particular, I demonstrate that a three-link robot can learn to lift heavy payloads when provided with a sequence of two or three intermediate configurations. Such sequences represent an initial solution that works with no payload but requires modification to solve the true task. The key differences between this framework and the usual trajectory-based methods are at least threefold: First, the initial kinematic path comes from "learning from demonstration" or from prior knowledge supplied by a coach, rather than from an explicit planning phase. Second, unlike most "demonstration" trajectories, the initial solution is relatively crude with few details in terms of both positions and timing. And third, the kinematic path is meant to be a starting point, not for tracking or for tuning, but rather for learning a related solution that exploits dynamics.

The initial solution, along with a measure of performance, forms the basis of a motor learning problem that can be solved by RL methods. The main result of Chapter 3 involves not the success or failure of a particular learning algorithm, but rather the structure of the control policies before and after learning. I show that coordinated movement is related to motor "synergies" that actively couple individual joint motions by way of the robot's control system. Moreover, without such coupling the robot becomes considerably impaired and unable to lift all but the lightest payloads.

One observation made in Chapter 3 is the existence of two qualitatively different solutions to the weightlifting task. These solutions differ in terms of the observed movement as well as the maximum payload achieved. In Chapter 4, I provide a theoretical analy-

sis of these solutions and trace their differences back to the velocity-dependent dynamics. The main result of Chapter 4 is a measure of *dynamic manipulability* that quantifies a robot's ability to accelerate its end-effector in various directions. This velocity-dependent measure is useful not only for analysis of observed trajectories, but also as a source of prior knowledge with which to seed a learning problem or trajectory planner. Moreover, a coarse-grained measure of the velocity-dependent effects can be calculated from manipulator positions alone, i.e., without knowledge of the velocity.

In Chapter 5 I return to the idea of a kinematic initial solution, yet in the context of a hybrid learning algorithm that combines RL with supervised methods. This algorithm implements a kind of steepest-descent search with two sources of gradient information, and one technical challenge is to balance the gradient information supplied by the supervisor with that of the RL component. As described in Chapter 5, one advantage of this approach is that prior knowledge can be used to design a controller, i.e., a supervisor, that performs sub-optimally yet provides performance guarantees while a reinforcement-based module learns an optimal control policy.

Figure 1.1 summarizes the organization of this thesis. Chapter 2 provides a multidisciplinary literature review, followed by the main results in Chapters 3–5. These latter chapters are largely self-contained with some connections made to the rest of the thesis. Informally, Chapter 3 is the motor control chapter, Chapter 4 the robotics chapter, and Chapter 5 the machine learning chapter. However, aspects of each discipline appear throughout the entire text, and the common thread is the notion of "exploiting dynamics" mentioned above. Chapter 6 makes the connections explicit and describes several possibilities for future work.

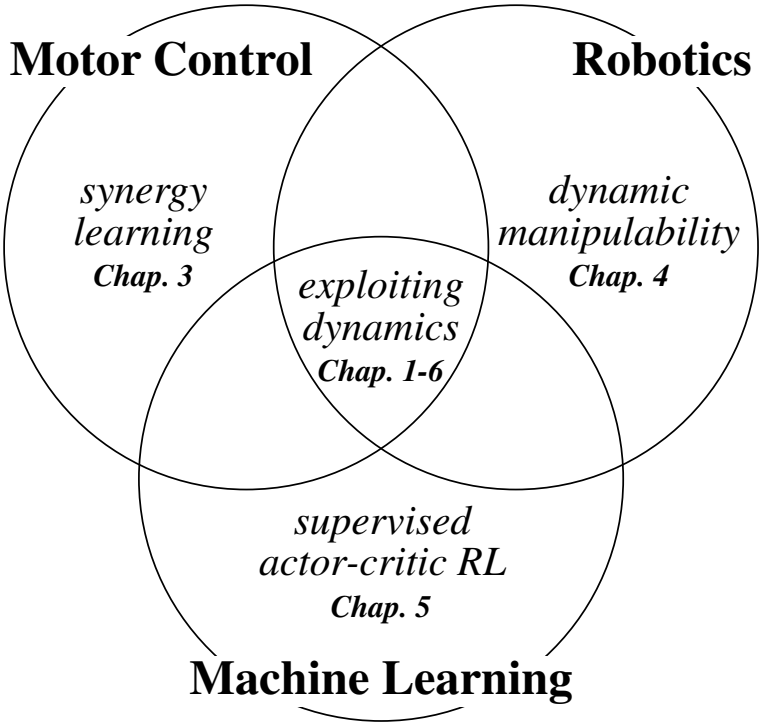

Figure 1.1. Venn diagram that summarizes the thesis organization.

CHAPTER 2

BACKGROUND AND RELATED WORK

The development of coordinated robots is a multidisciplinary endeavor, with a considerable amount of related literature. The goal of this chapter is to acquaint the reader with the terminology from several disciplines and to provide the necessary framework for describing the basic research questions, the tasks faced by robot manipulators, and the potential solution strategies to the "coordination problem." Consequently, the following review is somewhat general with additional details deferred to later chapters.

2.1 Human Motor Control

The research literature on human motor control is filled with general theories built-up from examination of specialized movements. This statement is intended to be both a criticism of the field and an acknowledgment of the difficult problems faced by this research community. Human motor control is such a broad topic that progress is best made by restricting one's attention to a specific class of movements. The criticism is that existing theories and hypotheses are often portrayed as competing rather than complementary. One goal of this research is to bring together several such hypotheses (e.g., motor programs and equilibrium-point control) as part of a coherent framework for robot motor skill acquisition. However, like other endeavors in movement science, the scope of results will be limited by the types of skills studied as part of this research.

2.1.1 Basic Types of Motor Skills

One way to classify motor skills is by the predictability of environmental effects. In particular, *closed skills* "can be carried out successfully without reference to the environ-

ment" [114]. Once the context has been assessed, the actor can execute a movement without additional external information (although internal information, e.g., kinesthesis, may be necessary). Closed skills typically involve "predictable requirements" [114], and intermittent feedback, e.g., from vision, may be necessary to realign the predictions with reality. *Open skills*, on the other hand, are relatively unpredictable and make use of frequent reference to the environment [114]. They require closed-loop control, whereas closed skills can be controlled in an open-loop fashion. Open skills can also proceed with or without advanced information; if enough information is available to support prediction, then the skill becomes closed from a functional perspective. For example, juggling is an open skill that becomes closed with practice, once the individual no longer requires frequent information about object locations.

Another common way to classify motor skills is by the termination of the movement [125]. At the extremes are *discrete movements* with a well-defined beginning and end, and *continuous movements* which could last for an arbitrary period of time. Examples of the former include swinging a racket and taking a step; examples of the latter include playing tennis and running. Somewhere in between are the *serial movements* such as handwriting, e.g., [148, 153], which continue for prolonged or indefinite periods of time yet are composed of a recognizable sequence of discrete movements. The examples described in this thesis deal primarily with closed skills and discrete arm movements, such as transport of an object from one point to another.

2.1.2 Basic Styles of Motor Control

In the mid 1960s, Richard Pew performed a series of experiments that illustrates the most basic styles of control which, to some extent, are incorporated in most theories of motor learning. Pew's seminal paper [113] describes an experiment where participants pressed one of two keys to accelerate a point toward one side or the other of an oscilloscope screen. The task was to regulate the position of the point as close as possible to the center of

the screen. Early in practice, control was entirely *closed-loop*, where subjects pressed one key, waited a relatively long period of time, and then initiated a feedback-based correction. Later in practice, participants "generated [an oscillatory] sequence of responses at a rate more rapid than could be controlled had [the subject] organized and executed each response as a separate unit..." [113]. Pew interpreted this strategy as a form of *hierarchical control*, where the oscillatory movement was predominantly *open-loop*, i.e., without feedback, and closed-loop corrections were used only at longer time scales (apparently when the average error exceeded some threshold).

Pew also described a "modulation" strategy used by all participants given a sufficient number of training sessions [113]. As with the hierarchical strategy, the movement was oscillatory and the control was open-loop. With the new strategy, however, errors were corrected not through intermittent feedback but rather by a gradual modulation of the duration one key was depressed relative to the other. Keele [78] interpreted this finding as evidence for the gradual modification of a *motor program*, which he defined as "a set of muscle commands that are structured before a movement sequence begins, and that allows the entire sequence to be carried out uninfluenced by peripheral feedback" [78, p. 387]. This frequently cited definition of motor program has been a source of controversy; specific points of contention were summarized by Rosenbaum [118] and are stated here as myths.

Myth 1: Motor programs are uninfluenced by peripheral feedback. Keele's definition places the emphasis on open-loop control, yet it does not preclude the use of peripheral feedback. Motor programs *allow*—rather than *require*—movement to unfold in the absence of peripheral feedback. Indeed, Keele also states "... movement control may become internalized.... There is no evidence, however, that performance is maintained without kinesthetic feedback" [78, p. 397].

Myth 2: Motor programs are structured like computer programs. Research in the 1960s was influenced by cybernetic theories [161] that focus on the information processing capabilities of animals and machines. The computer was a natural metaphor. The use of

"program" fosters, not so much a myth, but rather an unproductive dichotomy between those who reject computer metaphors for the brain and those who don't.

Myth 3: Motor programs directly encode muscle commands. This myth leads to excessive responsibility on the part of the program to represent fine details in the activation patterns of numerous alpha motoneurons. In reality, the program may encode abstract commands that trigger spinal oscillators or set gains in the alpha-gamma feedback loop, for example. Moreover, as the program executes in an open-loop fashion, the *direct* muscle commands could involve closed-loop mechanisms at some lower level of control.

Myth 4: Motor programs require a programmer. Motor programs are often criticized as executive-based mechanisms for coordination with no explanation of where the executive, i.e., the programmer, comes from. While there may be some truth to this myth, alternative theories that emphasize dynamical systems and self-organization suffer from a similar problem [82], i.e., they fail to explain the origin of the dynamics. For the purposes of motor coordination, Mother Nature is the ultimate programmer—supplying mechanisms for learning as well as innate structure that lift a bootstrapping process off the ground.

In Chapter 3, I build on the notion of motor program without subscribing to any particular approach, such as Adams' closed-loop theory [2] that emphasizes feedback-based corrections, or Schmidt's schema theory [124] that focuses on the open-loop aspects of control. (These theories are compatible in that they both allow for open-loop and closed-loop control and they both postulate the use of two memory "states," one for executing the movement and one for evaluating the result.) In this dissertation, I take the more general perspective suggested by Rosenbaum: A motor program is "a functional state that allows particular movements, or classes of movements, to occur [118, p. 109]." Thus, a motor program is simply a data structure, or memory, that supports the reproduction and generalization of a previously executed movement (cf. Bernstein's motor "engram" [21]).

While Rosenbaum's definition may be too broad as the basis for a theory of motor control, it does shift attention away from specific myths and toward general questions regarding

the nature of motor programs: How are the program outputs transformed into actions, and how are the lower levels of control coordinated [66]? How easily are motor programs modified [78]? How are biomechanical properties represented, and how are interactions with the environment dealt with and exploited [118]? How are new programs created ("the novelty problem") and how are existing programs generalized ("the storage problem") [124]? This thesis addresses each of these questions to some degree—not by postulating a theory of human motor control, but rather by developing specific mechanisms for robot motor control.

2.1.3 Basic Mechanisms of Coordination

Pew's regulation experiment contributed behavioral evidence for hierarchical organization as a *style* of motor control. However, one can also view hierarchical control as a *mechanism*[1] for coordination, with both theoretical and neurophysiological support (e.g., Bernstein [21] and Weiss [158], respectively). Brooks [33] described one possible neural hierarchy beginning with the (relatively) primitive limbic system, which generates biological and emotional needs that are transformed by the association cortex into goal-directed strategies. These strategies are fleshed out by the projection system (sensorimotor cortex, cerebellum, basal ganglia, subcortical nuclei) and eventually lead to movement, with missing details handled at the spinal and musculoskeletal levels of control. As demonstrated with the robot weightlifting task in Chapter 3, this thesis deals with a similar hierarchical implementation of goal-directed movement and assumes that basic needs and abstract goals are already available to a lower-level learning system.

Perhaps the most basic mechanisms for coordination should be attributed to the musculoskeletal system, with stereotyped dynamics from pendulum-like appendages, e.g., [100],

[1] In this thesis I distinguish "style" from "mechanism" in terms of explanatory detail. A style refers to a general principle or technique, whereas a mechanism refers to a relatively specific implementation by the neuromuscular system. For instance, closed-loop control is a style, whereas the gamma feedback loop is a mechanism.

and with servo-like behavior from viscoelastic and contractile properties of the muscle tissue, e.g., [149]. While such passive effects may play a key role for some activities (e.g., steady walking, jumping) the active participation of a control system is necessary for a more comprehensive account of coordination.

The most primitive form of active control is the open-loop reflex arc that directly maps stimulus to response (with possible modulation by supraspinal inputs). Sherrington's Nobel prize-winning research on the synapse led to a theory of coordination with such reflexes as the building blocks for prolonged movements [127]. Though Sherrington's "reflex chaining" hypothesis was subsumed by a more sophisticated taxonomy of movement primitives—including oscillators and servomechanisms, e.g., [61]—open-loop reflexes may still serve a crucial role at the extremes of motor proficiency. For instance, "the tonic neck reflexes appear to bias the musculature for movement in the direction of gaze," [55] thereby facilitating hand-eye coordination in infants and optimal force production in adults [55, 118].

Computational perspectives on motor coordination make frequent use of vector fields, oscillators, and other types of movement primitives as a mechanism for more skillful movements, e.g., [104, 123, 163]. The argument in favor of such primitives is straightforward; interactions between organism and environment are too complex for goal-directed movement to be constructed directly from momentary effector commands. Surprisingly, the utility of movement primitives for coordination is not without controversy. For instance, Turvey [146] suggests that movement primitives are insufficient to explain coordination and may even introduce unwanted complexity into Bernstein's problem: "Resolutions of the problem couched in terms of arranging fixed movement elements were dismissed by Bernstein. Reflexes, for example, were not elements of coordinated actions for Bernstein but, rather, elementary coordinated actions and, therefore, part of the problem of coordination rather than contributors to its solution" [146, p. 398]. This interpretation, although consistent with Bernstein's dismissal of Pavlov's conditioned reflex theory of coordina-

tion, may be incomplete with regard to the acquisition of new motor skills. Movement primitives—including reflexes—could qualify as what Bernstein called a "co-ordinating technique...based on past experience accumulated...by the genus" [21, p. 175]. In other words, innate movement primitives may play a role during learning even if they are unrecognizable in the final, coordinated movement.

In addition to movement primitives and hierarchies, internal models supply another important mechanism for coordination. Desmurget and Grafton [51] reviewed evidence that forward models enable closed-loop control when sensory feedback is inaccurate, delayed, or absent altogether. Such models could be used to extend Adams' closed-loop theory [2] to include fast movements without resorting to the open-loop style of control emphasized by Schmidt's schema theory [124]. However, Desmurget and Grafton also argue that "the motor command is not generated exclusively in real time" through error-correction with an internal feedback model [51]. For instance, anticipatory movements involve a predictive, *feedforward* motor command that precedes motion, and so at least a partial specification of some movements must be generated ahead of time. And while the term "model" often connotes an explicit representation of physical principles, such as the gravity term of a manipulator's equations of motion, one can also view a model as simply a mapping from motor commands to predicted movements (a forward model) or from desired movements to the motor commands needed to generate them (an inverse model). Far less is known about the nature of such mappings in the nervous system than, say, reflexes, although the use of internal representations has long been recognized as an important mechanism for motor control, e.g., [21, 65, 74, 78, 150, 164].

Though less of a mechanism and more of a principle, optimization is another aspect of coordination worth mentioning at this point. Clearly, optimization is necessary for highly skilled movements as in sports competitions. However, optimization also plays a crucial role in structuring otherwise ill-defined tasks—such as those involving redundant degrees of freedom—where optimality is secondary to the goal of completing the task at a sufficient

level of proficiency. A number of researchers have already made the link between optimization and human motor control, e.g., [56, 60, 88, 143], and such efforts beg the following question: "If optimality is a principle for motor control then what are the neuromuscular system's performance criteria?"

Many experimental studies tackle this question with (numerical) optimal control techniques along with a musculoskeletal model and a putative objective function. A good fit between simulation results and experimentally observed data provides some evidence that the putative objective function is of biological importance. (However, one has to draw conclusions carefully since more than one objective function could yield identical results.) For instance, Pandy *et al.* [110] examined several performance objectives for the task of rising from a chair. Their simulation suggests that individuals minimize some criterion related to energy prior to seat liftoff, but afterwards they maximize some other criterion related to the smoothness of the control effort. The authors speculated that the chair acts as a constraint which breaks the overall task into two sub-tasks with fundamentally different objectives. While knowledge of the performance criterion may tell us a great deal about the solution to a particular motor task, the question regarding the choice of a specific criterion may be less important than the actual use of optimality as a guiding principle. In other words, the optimization principle drives skill acquisition as long as a reasonable choice is made from many possible performance objectives.

2.1.4 Stages of Learning

As mentioned in Chapter 1, too little is known about biological motor control to provide us an obvious direction when designing robots that learn to solve motor tasks on their own. However, several researchers have offered relevant speculations about the stages of motor learning. For instance, Fitts and Posner [59] described three stages of skill learning, beginning with a "cognitive" phase when an adult tries to match early attempts at a new skill with his or her understanding of the motor task. Learning by demonstration or by

instruction from a coach is most effective during this phase of skill learning. Afterwards, performance improves gradually during the second, "associative" phase as the verbal (i.e., cognitive) aspects of learning drop out and as subtle adjustments are made in response to practice [125]. The final, "autonomous" phase is the result of extended practice and is characterized by even less cognitive influence and by relatively little interference from other activities and distractions.

This description of adult skill learning corresponds nicely to the progression of robot motor learning already mentioned in Chapter 1 and further detailed in Chapter 3. Another descriptive analysis of motor learning—one that comes closer to mechanisms—was offered by Bernstein in 1940 [21]: When a novice first attempts a new skill, excess degrees of freedom (DOF) are "frozen" in order to simplify the control problem. Later, these same DOF are gradually released, in two stages, as performance improves on the simplified task and mastery of additional DOF becomes a feasible goal. The first stage of release "corresponds to the lifting of all restrictions, that is, to the incorporation of all possible degrees of freedom.... The second, highest stage of co-ordinational freedom corresponds to a degree of co-ordination at which the organism is not only unafraid of reactive phenomena in a system with many degrees of freedom, but is able to structure its movements so as to *utilize entirely the reactive phenomena* which arise" [21, pp. 108].

Human gait supplies a familiar example of Bernstein's three stages of motor learning. In particular, Bril and Breniere [30] found that infants utilize a wide base of support (BOS) during their first few months of walking, but by two years of age the BOS narrows as the child's gait becomes more adult-like. Improved lateral control follows from a wide BOS, and so infants initiate gait by rocking side-to-side with locked knees. The authors draw a contrast to gait initiation in adults, where control in the sagittal plane requires mastery of relatively many DOF. By locking their knees, infants freeze degrees of freedom when they first learn to walk, and later release those DOF as the base of support narrows [30].

Moreover, by three years of age children also make use of anticipatory shifts in body mass as a way to exploit dynamics when initiating gait [87].

Similarly, as part of a longitudinal study of infants learning to reach, Berthier *et al.* [23] found that infants facilitate initial progress by locking degrees of freedom in the distal part of the arm while performing a trial-and-error search through a space of possible solutions. Infants also guide this search by limiting the peak movement speed as well as the number of hand starting locations. As the child becomes more proficient, degrees of freedom are released and the distal musculature is used to pre-orient the hand for a more efficient grasp [23]. And after many years of experience, humans further refine their reaching movements to maximize smoothness while accounting for limb dynamics [147]. The initial freezing or coupling of redundant degrees of freedom, therefore, may be a suitable starting place for robot motor learning that eventually leads to movements that exploit, rather than cancel, the robot's intrinsic dynamics.

2.2 Robotics

For robot manipulators, the classic solution to a movement problem involves the construction of a kinematic trajectory that avoids obstacles and singularities. One early approach by Lozano-Perez [93] made use of path planning in *configuration space*, where obstacles and safe paths are represented in a more manageable way. The underlying assumption of such methods is that the robot's intrinsic dynamics are something to compensate for, instead of something to exploit. In other words, typical path planners and trajectory generators use kinematic models and leave it to the control system to deal with the complex nonlinear dynamics that arise during the trajectory *tracking* part of the overall movement problem [128]. Even for optimized trajectories (e.g., [105, 120]) imprecise dynamic models often lead to overly conservative acceleration constraints and, therefore, to sub-optimal movements [46].

The strength of current control schemes for industrial robots lies in the way they solve separate planning and tracking sub-problems with little or no interaction between the two levels of control. Once a suitable trajectory is found, or supplied, the tracking problem can be solved using control engineering methods with desirable stability guarantees. However, the cost of this approach is inefficiency and a failure to make full use of a robot's capabilities. The most successful applications of such robot control methods involve highly predictable, repetitive tasks that often require specialized setup, thereby reducing the main advantages, i.e., flexibility and adaptability [29], that robots have over other forms of automation. Thus, the standard approach to robot control suffers from inadequate "communication" or interaction between the methods that solve the planning and tracking sub-problems. One goal of this thesis is to demonstrate the use of trial-and-error learning methods as a way to handle the planning and tracking problems simultaneously.

2.2.1 Manipulator Dynamics

Before moving on to a survey of robot control methods, in this section I first summarize the main results regarding the dynamics of robot manipulators. My survey is limited to rigid robots with revolute joints and ideal actuators; see the review by Sage *et al.* [119] for a treatment of flexible joints and non-trivial actuator dynamics.

For an open kinematic chain of n links, the equations of motion can be derived using either of the Euler-Lagrange or iterative Newton-Euler formulations, e.g., [135] or [47], respectively. The dynamic equation can be expressed as

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\dot{\mathbf{q}}) + \boldsymbol{\tau}_d, \tag{2.1}$$

where $\boldsymbol{\tau}$ is an $n \times 1$ vector of joint torques and $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$ are $n \times 1$ vectors of joint positions, velocities, and accelerations, respectively. In Eq. (2.1), $\mathbf{M}$ is an $n \times n$ *mass matrix* that captures the configuration-dependent inertial properties of the robot. The mass matrix is symmetric and positive definite and, therefore, invertible, except for degenerate

idealized models of open-chain manipulators. Moreover, both $\mathbf{M}$ and $\mathbf{M}^{-1}$ are bounded from above and below—an important property for establishing stability of some control methods, e.g., [46].

In Eq. (2.1), the vector of torques represented by $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is quadratic in $\dot{\mathbf{q}}$, with terms of the form $\dot{q}_i^2$ accounting for centrifugal forces and terms of the form $\dot{q}_i\dot{q}_j$, $i \neq j$, accounting for Coriolis forces. Simple control schemes that ignore such forces typically perform well only at low velocities, where the quadratic terms are small, whereas more sophisticated methods often utilize a model to compensate for velocity-dependent forces. The $n \times 1$ vectors $\mathbf{G}$ and $\mathbf{F}$ represent, respectively, the joint torques due to gravity and friction. Finally, $\boldsymbol{\tau}_d$ is an $n \times 1$ vector of disturbance torques that accounts for joint flexibility, parameter mismatch, actuator nonlinearities, unmodeled friction effects, etc. Typically, assumptions are made about the bounds on the disturbance torque, although the precise form of $\boldsymbol{\tau}_d$ is unknown.

2.2.2 Manipulator Control

Perhaps the most straightforward approach to position control of a robot manipulator is to use independent servos that work to reduce the Cartesian errors attributed to each joint. This "conventional" approach [94] or "inverse Jacobian controller" [47] or "resolved motion-rate control" [160] involves the mapping of errors in Cartesian space, $\delta\mathbf{X}$, to errors in joint space, $\delta\mathbf{q}$:

$$\delta\mathbf{q} = J^{-1}(\mathbf{q})\delta\mathbf{X}. \tag{2.2}$$

In Eq. (2.2), $\delta\mathbf{X}$ and $\delta\mathbf{q}$ are actually differentials of the same dimensionality and J is the $n \times n$ *Jacobian matrix* that relates the instantaneous change in joint position to the configuration-dependent change in Cartesian position of a point near the tip of the manipulator (usually the wrist or the end-effector, e.g., gripper, welding torch, paint nozzle). The Jacobian is well-defined given the robot's kinematic specification—often expressed using Denavit-

Hartenberg notation [50]—although the inverse does not exist for singular configurations such as when the arm is fully extended. As long as the measured $\delta\mathbf{X}$ is small enough, Eq. (2.2) yields a good approximation of the joint servo error at the start of each control cycle. Moreover, the servo controller can track arbitrarily long paths if an interpolation scheme is used to break a relatively coarse-grained path into segments where each $\delta\mathbf{X}$ is small.

When $\delta\mathbf{X}$ and $\delta\mathbf{q}$ have different dimensionality, the inverse Jacobian is undefined and Eq. (2.2) is often modified to use the *pseudoinverse*, $J^\dagger$, instead. As above, assume that $\mathbf{q}$ is an $n \times 1$ vector, but now suppose that $\mathbf{X}$ is an $m \times 1$ vector with $n \neq m$. Then J is an $m \times n$ matrix and the $n \times m$ pseudoinverse is defined as

$$J^\dagger = \begin{cases} (J^{\mathrm{T}}J)^{-1}J^{\mathrm{T}}, & \text{if } n < m \\ J^{\mathrm{T}}(JJ^{\mathrm{T}})^{-1}, & \text{if } n > m. \end{cases}$$

Of particular interest for this thesis is the case for redundant manipulators, i.e., for $n >$ m when there are more independent joint motions than there are degrees of freedom that need to be controlled in Cartesian space. For redundant manipulators, the pseudoinverse gives the best solution in a least-squares sense. That is, $\delta\mathbf{q} = J^\dagger\delta\mathbf{X}$ gives the joint-space differential that minimizes $\|\delta\mathbf{q}\|_2$. However, $J^\dagger\delta\mathbf{X}$ describes just one of an entire family of joint motions that yields the desired Cartesian motion. More generally,

$$\delta\mathbf{q} = J^\dagger\delta\mathbf{X} + (I - J^\dagger J)\delta\Theta, \tag{2.3}$$

where $\delta\Theta$ is an arbitrary joint differential and $(I - J^\dagger J)\delta\Theta$ is a projection into the null space of J, e.g., [70]. Thus, $\delta\Theta$ can be chosen to achieve a secondary objective without disturbing the primary objective represented by $\delta\mathbf{X}$.

Although simple and potentially effective, the inverse Jacobian controller has several drawbacks. First, the approach is best-suited for trajectory planning in Cartesian space,

where interpolation could place the manipulator near a singularity or outside its reachable workspace, or both. Second, a drawback for controller design is that fixed gains will result in a dynamic response that varies with arm configuration [47]. Third, and most relevant for this thesis, is that the approach ignores the manipulator dynamics altogether; fast movements are not only inefficient but also inaccurate or unstable.

Other conventional, though more sophisticated, strategies for controlling rigid manipulators are based on the *computed torque* or *inverse dynamics* method, which Kreutz [85] showed to be a specific form of *feedback linearization*. The essential idea is that by examining the structure of Eq. (2.1) one can devise an inner feedback loop that linearizes the system, i.e., compensates for the coupled, nonlinear manipulator dynamics. The result is a new system in a canonical linear form which, in turn, can be controlled by an outer loop designed using standard linear control techniques. In particular, the control for the inner loop is given by

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})u + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\dot{\mathbf{q}}),$$

where u is the output from the outer loop control law. If one ignores the disturbance torque $\boldsymbol{\tau}_d$, then substitution of this inner control law into Eq. (2.1) yields

$$\ddot{\mathbf{q}} = u.$$

Thus, u can be interpreted as the desired acceleration of the manipulator, i.e, the acceleration needed to track a desired trajectory, with possible compensation for position and velocity errors.

The theoretical assumptions underlying the computed torque method are the invertibility of mass matrix and the independent control of each joint [85]. In practice, the computed torque method also assumes that $\mathbf{M}$, $\mathbf{C}$, $\mathbf{G}$, $\mathbf{F}$, and $\boldsymbol{\tau}_d$ are known. Uncertainty in any of these quantities requires the use of additional techniques to ensure stability of the overall control

method. For instance, adaptive control techniques can be used to design an adaptation law that adjusts the current estimates of **M**, **C**, **G**, and **F** and that guarantees convergence in the error dynamics [109]. Robust control techniques, on the other hand, are geared toward the design of controllers that yield acceptable performance given estimates of the parameter uncertainty [1]. Both approaches share some similarities, although robust methods typically make use of a *fixed* inner control loop that gives a nominal linearization of the robot dynamics; an outer control loop then deals with robustness [119, 131] as well as trajectory tracking errors.

Many results on adaptive and robust control of manipulators are of theoretical interest but, at present, are not of practical significance because they fail to address the issue of performance at a realistic task [35, 119]. As emphasized throughout this thesis, one general way to improve performance is to exploit the dynamics of the task and the robot. *Passivity* approaches are particularly interesting in this regard because they exploit physical properties in order to reshape a system's potential energy such that the minimum falls near the desired equilibrium point [119]. For instance, one can exploit the dissipative nature of friction to get the beneficial effects of higher servo gains [132]. Unfortunately, passivity approaches only address the design of stable controllers with favorable convergence properties; the issue of task performance is still left to a separate trajectory planning algorithm.

For tasks that involve interaction with the environment—beyond pointing and gesturing tasks—the force imparted by a robot on its environment must be controlled in addition to its position. Force control is important not only for efficient transport and manipulation of objects but also for safety purposes related to position control. During a part insertion task, for instance, sensor error could result in excessive force by a "position-only" controller and, therefore, damage to the part. As summarized by Zeng and Hemami [167], fundamental approaches to force control differ primarily in the way force feedback is used to adjust (or replace) position and velocity feedback. In effect, force control methods either augment

the position trajectory with desired forces, alter the apparent dynamics of the robot (e.g., impedance control [69]) or both.

Although this dissertation deals with some aspects of object manipulation, the class of motor tasks studied during this project does not involve stationary objects as in applications such as grinding and part-mating. Thus, explicit use of force control is beyond the scope of this research. Instead, I "finesse" the force control problem in a manner similar to the switching and tuning approach used by Narendra and colleagues [42, 107]. Their approach employs multiple adaptive models, one for each task that the robot may face in terms of end-effector load or other variable that changes the dynamics without changing the structural form of the adaptive models. The benefit of this scheme is an improved transient response for trajectory tracking because the switching controller selects the best candidate model for subsequent tuning. Similarly, in Chapter 3 I assume that object manipulation can be treated as a position control problem for a robot with variable inertial properties. In effect, the set of objects possibly manipulated by the robot induces a set of similar, yet distinct, coordination problems over which the control system must generalize.

2.2.3 Learning and Control

Unlike the methods reviewed thus far, optimal control theory, e.g., [34, 81], provides a framework for solving the trajectory planning problem while accounting for manipulator dynamics and task constraints simultaneously. While extremely effective with accurate models, optimal control methods yield solutions that tend to be brittle in the presence of disturbances (i.e., unmodeled effects) as well as modeling errors. One possible way to deal with such brittleness is to incorporate a measure of uncertainty as part of the performance criterion. (Lin and Brandt [91] showed one way to do this, although their work is in the context of robust controller design for trajectory tracking problems.) Learning methods are particularly effective in this regard because the performance criterion need not be expressed in a concise analytical form; robustness can come from heuristics that downgrade

the performance evaluation whenever the learning agent enters an unfavorable situation. Another advantage of such methods is the ease with which the robustness requirement can be implemented in a non-uniform way over the system's range of operation.

Learning control provides another approach to manipulator control that deals with uncertainty [7, 11, 45, 71]. Such methods take advantage of the repetitive nature of many robot motor tasks by introducing a *betterment process* [7] that improves upon the robot's performance during the previous repetition. Although the focus of learning control is the use of a feedforward control law that suppresses repetitive disturbances, feedback remains a necessary component for both robustness and learning (cf. feedback-error learning [63]). Unlike optimal control methods that require accurate models, learning control algorithms require only models that are good enough to derive a meaningful servo error for the update of the feedforward command [11].

With learning control, motor dexterity is gained not through the use of feedback mechanisms, as with adaptive control, but rather "through the use of a feedforward control action which is stored in memory and is retrieved as the task is executed" [71]. However, like the other control methods discussed above, learning control assumes that the motor task is represented as a prespecified trajectory. The perspective taken in this thesis is that dexterity depends upon all three components: feedback control for robustness, feedforward control for anticipatory action in the absence of useful feedback, and (implicit) trajectories for coordinated movements that exploit dynamics.

2.3 Machine Learning

Over the past decade, one trend in the control engineering literature has been the increased use of techniques that incorporate aspects of "intelligent" decision-making [106, 117]. These techniques include expert systems, fuzzy logic, genetic algorithms, and artificial neural networks. Of particular interest for this thesis are the reinforcement learning (RL) methods studied by the machine learning and operations research communities. Such

methods are suitable for learning motor skills through ongoing practice when detailed models are unavailable. Since RL methods solve optimal control problems, they also implicitly solve Bernstein's problem using a form of optimization for redundancy resolution. As stated in Chapter 1, however, the practical success of RL depends a great deal on prior structure. After a brief introduction to RL, the remainder of this section summarizes research regarding two specific forms of prior structure: hierarchical architectures and hints from a teacher.

2.3.1 Reinforcement Learning

Modern RL algorithms have their origins in animal learning research, with particular emphasis on the notion of trial-and-error learning summarized by Thorndike [142, p. 244]:

> "Of several responses made to the same situation, those which are accompanied or closely followed by satisfaction to the animal will, other things being equal, be more firmly connected with the situation, so that, when it recurs, they will be more likely to recur; [and those associated with discomfort] will be less likely to occur."

For a learning agent, such as a robot, the essence of Thorndike's "Law of Effect" is that the agent chooses actions ("responses"), uses reward ("satisfaction") to evaluate their consequences, and associates ("connects") the best actions with specific states. But because the consequences of its decisions may not be fully predictable, the agent must choose actions that balance immediate gain with the possibility of high rewards in the future.

Most algorithms for trial-and-error learning fall near one of two extremes: those that take advantage of structure in the problem, and those that take advantage of structure in the solution. RL algorithms such as TD(λ) [137] and Q-learning [157] are particularly well-suited to take advantage of structure in the problem. Such algorithms use a value function to capture regularities in the observed rewards and state transitions that implicitly define the task. In its most basic, mathematical formulation, an RL problem is represented by a

Markov decision process (MDP), whose solution is an optimal policy that maps states to actions and that satisfies the *Bellman optimality equation*:

$$V^*(s) = \max_a \sum_{s' \in \mathcal{S}} \Pr(s'|s,a)\{R(s') + V^*(s')\}, \tag{2.4}$$

where $V^*(s)$ is the value of a state s, $R(s')$ is the expected immediate *reward* received in the next state, s', and $\Pr(s'|s,a)$ is the probability of transitioning to state s' when action a is executed from s. Eq. (2.4) is a recursive formula that asserts that the value of any state, when following an optimal policy, equals the expected total reward when the best action is chosen from that state. One advantage of RL over more traditional optimal control theory is that no state transition model is required in advance; the agent, or controller, improves over time as it interacts with, and gains experience about its environment.

Much research in RL has focused on extensions to this basic formulation, including discounted rewards, e.g., [138], factored MDPs, e.g., [27], semi-Markov decision problems, e.g., [140], and partially observable MDPs, e.g., [76]. The use of function approximation is one particularly relevant extension for robotics research. For instance, one could utilize an artificial neural network—with a finite number of parameters—as a way to approximate V^* over a high-dimensional, continuous state space, such as those involving the kinematic state of a multi-link robot. In contrast to reinforcement-based methods, where the learning algorithm receives a simple evaluation of its performance, function approximation typically involves *supervised learning*, where parameters are tuned based on more detailed instruction about the correct action or outcome. (In the motor learning literature, a similar distinction is made between *knowledge of results* and *knowledge of performance*, e.g., [31, 125].) Sutton [137] and others showed that predictions about expected rewards can supply the needed feedback for learning with function approximation when no other instructional data are available. More recently, a number of papers point toward a growing interest in the use of function approximation for representing the policy in addition to (or in place of) the value function [5, 13, 19, 37, 80, 83, 139, 162].

Along with their generalization capabilities, one advantage of function approximation techniques is the feasibility of incorporating prior, possibly inexact, knowledge. For example, Towell and Shavlik [144] constructed an artificial neural network that captured a body of knowledge expressed in first-order logic. The initial parameter values were then refined by a supervised learning algorithm, with observed training data as the input. In a similar fashion, the work presented in Chapter 3 demonstrates the use of function approximation to represent movement primitives as a form of prior knowledge. These primitives are built in a manner analogous to the cognitive phase of human motor learning, where instruction plays a role in shaping the initial solution to a motor task. (See Section 2.1.4.) But rather than tune the movement primitives through supervised learning, RL provides the mechanism for mastering redundant degrees of freedom.

One criticism of RL algorithms is that they typically ignore structure in the solution by treating the policy as a featureless mapping from states to actions. This criticism is partial motivation for the other extreme of trial-and-error learning, where methods for direct policy search, e.g., [5, 13, 19, 102], do away with intermediate representations of the policy and take advantage of constraints placed on the solution. For instance, evolutionary algorithms, e.g., [102], reinforce regularities in the parameterized policy (the "genetic material") by restricting "reproduction" to the fittest members of the population. And direct methods for function optimization, e.g., [141], make the most of continuity by biasing their search toward promising regions of parameter space. However, such methods ignore structure in sequential decision problems by making policy adjustments on a trial-by-trial basis after long-term rewards are known, rather than on a per-action basis. Ideally, trial-and-error learning algorithms should take advantage of the strengths of both policy search and value-based methods, e.g., [13]. Chapters 3 and 5 demonstrate the use of each kind of trial-and-error learning separately, and in Chapter 6 I discuss the possibility of combining both methods by using a learned value function to bias a direct search for the optimal policy.

2.3.2 Hierarchical Architectures

Twenty years ago Peter Greene [66] used the words "patchwork of methods" to characterize the way humans control arm movements as well as the way one should design effective methods for coordinating redundant degrees of freedom in robotic arms. Although a "single powerful technique" [66] may be appropriate for some machine learning problems, in this thesis I pursue the use of a combination of strategies for learning complex motor skills. For instance, the robot weightlifter in Chapter 3 demonstrates the use of RL at two different levels of a hierarchical control architecture, where each level is viewed as an opportunity to add structure to the learning problem rather than simply as a means for dealing with computational complexity.

With mobile robots, a "patchwork of methods" is a common design strategy for artificial intelligence researchers. For instance, Brooks' subsumption architecture [32] makes use of multiple "layers of competence" that decompose the control task into classes of simple behavior modules with different goals and different ways of processing the available sensor data. Indeed, any modular architecture qualifies as a patchwork as long as the modules, i.e., the patches, utilize distinctive techniques for control or learning. For control, another possibility is a three-layer architecture [62] with a deliberative planner at the highest level, a "sequencer" that executes the current plan, and a collection of reactive controllers (behaviors) at the lowest level. For learning, a bottom-up approach [136] can lead to complex team strategies from individual skills and primitive behaviors, with each layer employing the best machine learning algorithm for the subtask. More generally, patchwork approaches serve as a counterpart to *tabula rasa* learning, with behavior-based learning as the dominant paradigm for mobile robots [8].

As mentioned previously, the dominant approach for robot manipulators involves the use of a model—either constructed, e.g., [4, 47], or learned, e.g., [63, 71, 152]—that "compensates" for the robot's nonlinear dynamics and converts the control problem to one of tracking a desired trajectory with an approximately linear plant. One could view

the trajectory-tracking approach as hierarchical with a planner at the highest level and two subordinate control loops, as described in Section 2.2.2. However, such methods are *flat* with regard to the desired movement; the control hierarchy hides detail in the calculation of the torque signal, yet a single sub-system (the planner) is responsible for all the detail in the position command. The only details filled in by the lower-level sub-systems are those related to disturbances away from the specified trajectory. This differs from the approach taken in Chapter 3, where the control hierarchy provides abstraction in the way a coordinated movement unfolds, cf. Greene's "ballparks" [65], with each level responsible for some degree of detail in the solution to the movement problem.

Several recent efforts have made progress at formalizing the link between hierarchical abstraction and RL. For instance, Parr and Russell [111] used a hierarchy of finite state machines where some states represent choices subject to learning and others denote subroutine calls to additional machines. Dietterich [52] showed that a tree-like data structure can be used to decompose a policy into context-independent and context-dependent subroutines and to represent the corresponding hierarchical value function. And Sutton *et al.* [140] introduced *options* as a "minimal extension" of the MDP framework that allows for temporally extended actions. (See [17] for a detailed review.) At present, these formal approaches to hierarchical RL are largely of theoretical interest and applications have been limited primarily to problems with relatively small sets of discrete states and actions. Although the efficacy of such methods for robot motor learning is a relevant issue for this thesis, I instead consider algorithms that are more *ad hoc* with regard to hierarchies, yet general with regard to the design of motor learning problems.

2.3.3 Hints from a Teacher

In addition to function approximation and hierarchies, one can structure a machine learning problem through "hints" given by a teacher or supervisor. The intuition behind this general approach is familiar to RL researchers, who often design learning problems

with a carefully constructed reward signal meant to encourage the desired agent behavior. In effect, the designer specifies all of the hints beforehand, although there is growing interest in methods for incorporating teacher hints in an online, incremental fashion. One natural way to do this is to modify or *shape* the reward signal, e.g., [53, 97, 108, 159]. Such methods are analogous to the way animal trainers distribute well chosen rewards (or punishments) to lead the learning agent, i.e., the animal, through a progression of learning problems, beginning with ones that are relatively easy to solve.

In lieu of an explicit teacher, another way to include supervisory information is to design controllers that take advantage of prior knowledge. For instance, Randlov *et al.* [116] and Perkins and Barto [112] used control-theoretic techniques to design closed-loop feedback controllers with desired performance guarantees. Huber and Grupen [72] used a similar approach for a quadruped robot that learned to walk by composing relatively primitive behaviors. Essentially, each of these examples used closed-loop control policies as options [140] that simplified learning at a higher level of decision making.

A third possibility for combining supervisory information with RL algorithms is to allow the teacher to suggest actions for the learning agent to choose from. In effect, the teacher exerts indirect control of the environment by influencing the exploratory behavior which is characteristic of trial-and-error learning algorithms. Examples of this approach include ASK FOR HELP [43], RATLE [96], the mentor framework [115], and LBW [159]. These methods all represent the policy implicitly through a learned value function, whereas the approach described in Chapter 5 makes use of an explicit data structure for the policy that enables direct learning, i.e., cloning [14], of the teacher's unknown policy.

In recent years, work related to "teacher hints" has also gained popularity with the intelligent robotics community, though primarily under the rubric of "imitation learning," e.g., [98, 122]. Imitation, now considered a sign of intelligence, was once viewed as merely a form of replay or simulation of intelligent behavior [36]. Although there exists no widely accepted definition of imitation, it is nevertheless accepted that imitation comes in many

different kinds involving varying degrees of intelligence. For instance, Byrne and Russon [36] draw a distinction between "action" and "program" levels of imitation. At the action level, the observer imitates the "surface form" or idiosyncrasies of behavior, possibly as a component of social interaction. Whereas at the program level, the observer copies the "organizational structure" of behavior as a means for acquiring new skills. In any case, the communication of novel behavior appears to be a crucial feature of imitation [36], as opposed to, say, "emulation" whereby the actions of the demonstrator are used to communicate goals to the observer rather than the behavior for achieving those goals.

Interestingly, approaches related to imitation learning for robot skill acquisition have a long history with industrial robots. In particular, before the availability of inexpensive computers and subsequent robot programming languages, industrial robots were commonly programmed with handheld *teach pendants* that were used to guide a manipulator through a sequence of configurations [47, 49]. Such efforts, which now utilize haptic interfaces in addition to teach pendants, are more appropriately called "teaching by showing," "learning by watching," "programming by demonstration," or "learning from demonstration" [10, 12, 86, 101, 121, 130]. In contrast with these terms, "imitation" connotes human-like intelligence on the part of the observer as well as the demonstrator.

As applied to robot assembly skills, learning by demonstration typically involves no adaptation or learning per se. Instead, a demonstrated trajectory (of positions, forces, or both) is digitized and heuristics are used to segment that trajectory into a symbolic description of the desired skill. For instance, Asada and Izumi [10] used human demonstration to solve an object placement task by building a sequence of "move" and "push" commands that account for both positions and forces, respectively. More elaborate systems often utilize video-based hardware to acquire the human demonstrator's movement trajectory. This allows for a more natural and potentially faster communication of the desired behavior than is normally possible with teach pendants and their more modern counterparts. For example, Atkeson and Schaal [12] and Miyamoto *et al.* [101] used a combination of visual

input, numerical optimization, and both inverse kinematics and inverse dynamics models to acquire dynamic motor skills for a seven-DOF manipulator. Learning, although a part of both projects, was used primarily for system identification purposes and for relatively minor tuning of a trajectory-based solution to the motor task.

Most examples of robot learning by demonstration use "learning" metaphorically as a way to describe the role of the demonstration data. Few examples take advantage of the exploratory behavior of RL methods—perhaps due to the strong theoretical assumptions made by more formal approaches, e.g., [115]. One exception comes from Kaiser and Dillman [77], who used human demonstration data to seed an actor-critic architecture for learning slow kinematic skills. Similarly, for a pendulum swing-up task Schaal [121] combined RL with demonstration data and compared the effects of seeding the policy, the state-transition model, or both. The key difference from this dissertation is that with each of these examples, the seed data were relatively detailed and RL was used primarily to overcome the lack of a model. In effect, RL was used to tune a given solution rather than to discover a new one consistent with the initial data.

CHAPTER 3

LEARNING AT THE LEVEL OF SYNERGIES

The basic premise of this research is that motor coordination is not simply a by-product of trial-and-error learning, but rather that coordination emerges when "regularities," as Bernstein put it, are part the learning technique. In this chapter, I show that given a suitably designed class of policies, with biologically motivated structure, a simple search algorithm is then able to solve an otherwise complicated motor task. Moreover, by manipulating the class of policies I also show that the best solutions involve "synergies" that cause individual degrees of freedom to act as a single functional unit.

3.1 The Level of Synergies

As suggested in Chapter 1, the term "coordination" is relatively informal, referring to a more intuitive concept than "control." Whereas mechanisms of control are specific to the available inputs, outputs and communication pathways, coordination instead deals more with the style and organization of these lower-level mechanisms. In turn, the mechanism for coordination is attributed to motor synergies that implement "the *organization of the control* of the motor apparatus" [21]. Bernstein's original writings about synergies remain to be translated into English [68], although an account geared toward a wide audience appeared in a "lost" book [22] published for the first time in 1996, as an English translation from the original Russian manuscript.

In his book, Bernstein described several levels involved in the construction of movements, some of which deal with muscle tone, spatial relationships, and goal-directed actions [22]. Of particular relevance for this thesis is Bernstein's "level B" or level of

"muscular-articular links" or, simply, the level of synergies. At the level of synergies individual degrees of freedom, i.e., muscles and joint motions, are organized into functional units that solve a motor task as a whole. Synergies are similar to "coordinative structures" [55, 79, 145] which combine muscles into groups "that are constrained to act as a single unit" [79]. From a more control theoretic perspective, one can think of synergies as intelligent nonlinear controllers that exploit the dynamics of multi-input, multi-output systems.

More recently, a number of researchers, e.g., [73, 90, 99, 148, 151], used the notion of synergies to explain observed reproducibility of endpoint motion despite the relative variability of individual joint movements. Nevertheless, very little is known about how synergies develop, since they are difficult to tease apart from the dynamics they exploit. Indeed, statistically conclusive evidence for the existence of synergies in humans is apparently provided by just one series of experiments [90] that examined the load-sharing patterns of several fingers during a simple isometric task, i.e., a task with trivial (external) dynamics. Observed variability in force produced by each finger individually was less than the variability in the total force. In this scenario, the experimenters were able to attribute the observed results to the neuromuscular system, rather than to some effects of the limb dynamics.

3.2 Robot Weightlifting

Most theories regarding the mechanisms of human motor coordination rely a great deal on inferences made from observed *external* behavior. Studies involving artificial systems have the advantage of a relative abundance of information from the *internal* control system. In the remainder of this chapter I demonstrate how this sort of information can shed light on the nature of motor synergies. More specifically, I describe several experiments with a robot motor task based loosely on Olympic weightlifting—an extreme example of a motor skill that requires an athlete to exploit dynamics such as gravity, inertia, elasticity, and so on.

For the robotics version of the weightlifting task, I developed a three-link manipulator comparable to the acrobot-like examples studied by the machine learning and control engineering communities. For example, Spong [134] showed that for the two-link acrobot with torque generated only at the second joint, a switch between two controllers is sufficient to bring the robot from its stable, downward equilibrium to its unstable, upright configuration. Spong's "swing-up" controller pumps energy into the system until the robot enters the estimated controllability region of a "balance" controller that regulates about the goal. The robot weightlifting task differs from the acrobot in at least several ways: (1) In contrast to control engineering solutions, I assume no detailed model of the robot dynamics. (2) The extra link for the weightlifter introduces additional complexity to the problem. (3) The weightlifter is underpowered but not underactuated; while underactuated systems may be harder to control, they have the benefit of a smaller space of possible controls that must be searched by a learning algorithm. (4) Joint constraints and obstacles for the weightlifter render the standard "swing-and-balance" solution ineffective.

Figure 3.1 shows a simulation of the robot in several configurations. (Section 3.5.3 describes work with a real robot.) This robot was modeled as a three-link frictionless pendulum with each link having length 1 m and and mass 1 kg. At each joint, torque was limited to $[-50,50]$ Nm, and at the middle and distal joints the range of motion was confined to $[-150,150]$ degrees. At the proximal joint, movement was also limited to a range of 300 degrees, to avoid contact with the robot's simulated support structure. The equations of motion were generated iteratively using the Newton-Euler method [154] and solved numerically by Euler integration with a step size of 0.001 s. The robot was motionless at the start, and the goal was defined as a six-dimensional hypercube with velocity limits of ± 28.65 deg/s (± 0.5 rad/sec) and with position limits of ± 5 degrees centered on the goal configuration. At the end of each trial, i.e., when the robot's state entered the goal region, performance was quantified as the total integrated torque magnitude, except on those trials where the lifter exceeded its range of motion, contacted an obstacle, or failed to reach the

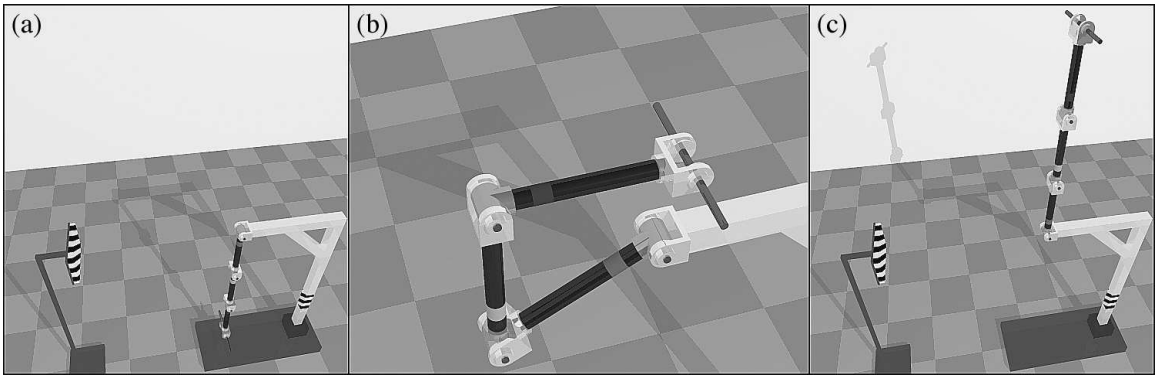

Figure 3.1. Simulated three-link robotic arm in several configurations with no payload: (a) start, (b) via point, and (c) goal.

goal within five seconds, in which case the trial was terminated with the maximum cost of 500 Nm·s. In summary, the task was designed as a minimum-effort optimal control problem with both torque and kinematic constraints.

3.3 Related Work

Optimal control theory provides a number of techniques for solving dynamic optimization problems such as the weightlifting task. For instance, Wang *et al.* [155] used a gradient-based method starting from an initial feasible path to more than triple the recommended payload capacity of a Puma industrial robot. As mentioned previously, however, such methods usually require precise models (or system identification) and protracted off-line computation before the robot attempts its first lift. In this work I make no assumptions about the availability of a detailed model. Instead, I borrow several human motor learning strategies as a way to design a structured policy that achieves coordination by means of a small number of simple motor commands.

Others have demonstrated the utility of simple motor commands for achieving control of relatively complex dynamic systems. For example, Barto *et al.* [16] used equilibrium-point controllers, with nonlinear damping, to demonstrate a cerebellar learning model for control of a simulated arm. Simple "pulse/step" commands were used to set the equilibrium

points in an open-loop fashion, and their approach differs from more standard theories in at least two important ways. First, the pulse and step specify just two equilibria, and complex motion emerges from the dynamics of the arm together with the modeled spinal reflex mechanisms, rather than from some unspecified source of trajectories. Second, the nonlinear damping enables rapid initial movement followed by slow drift to the equilibrium point. Control is geared toward fast movement to an effective endpoint, rather than the true equilibrium point, but with little or no oscillation. Thus, dynamics are exploited for the purposes of simplified control, rather than for optimization as with the kind of motor synergies described throughout this thesis.

Huber and Grupen [72] used a small set of "basis" controllers to learn a variety of walking gaits for a four-legged robot. These controllers guaranteed convergence to a discrete set of equilibria, thereby allowing a higher-level reinforcement learning system to treat the continuous control problem as one discrete in time, state, and action. Their approach is hierarchical, not only in the sense that low-level controllers are switched by a high-level learning system, but also in the sense that controllers run concurrently, with subordinate controllers not allowed to interfere with superordinate ones. One key distinction between their approach and the framework in this chapter is that the low-level controllers remained fixed throughout learning, whereas the controllers described herein change by trial-and-error learning.

For a simulated diver, Crawford [48] also used a hierarchical control system with a reinforcement learning agent responsible for switching low-level controllers. In particular, Q-learning [157] was used to set the switch times for an open-loop sequence of "behavioral synergies" (such as "throw" and "pike"). The simulation used a complex dynamic model of a human diver, although the lower-level sub-system used inertial compensation to track prescribed trajectories independently for each joint. These trajectories used bell-shaped velocity profiles to connect a desired sequence of joint configurations. Thus, from the

perspective of the learning system the task was kinematic in nature, with the "behavioral synergies" canceling the dynamics to track the specified trajectory.

To solve a robot stand-up task Morimoto and Doya [103] used reinforcement learning at two levels of a hierarchical control system. At the upper level, coarse-grained inputs and outputs were used to specify sub-goal configurations for the lower level, with joint velocities left unspecified. The robot's continuous state space acted as input to the lower level, which optimized the parameter values for a nonlinear controller that included linear error terms as well. Tabular Q-learning was used at the upper level, whereas the lower level used a continuous version of temporal-difference learning. The approach is most similar to the one presented in this chapter, although Morimoto and Doya make no connections to biological motor control, such as those connections involving synergies and learning from demonstration.

3.4 Structured Policy Parameterization

My solution begins with a sequence of *via points*, i.e., intermediate configurations of the robot along a path to the goal. I regard each via point as the specification for a movement primitive that converges to the corresponding manipulator configuration. Such primitives are suggestive of an equilibrium-point style of motor control, e.g., [25, 58], whereby the endpoint of a movement is programmed and the spring-like properties of coordinated sets of muscles, i.e., muscle synergies, ensure convergence to that endpoint. The primary benefits of equilibrium-point control are that complex limb dynamics can be ignored and that higher motor centers need not be involved in the detailed activation patterns of numerous actuators. The drawback is that to explain complicated skills, especially those involving fast movements, equilibrium-point approaches give up their simplicity by requiring a *virtual trajectory*, or sequence of equilibrium points that induce the desired movement but are not targets for convergence. Equilibrium-point models do enable multiple degrees of freedom to work together, although these models alone do not account for synergies that

exploit dynamics in a way that Bernstein described as "mastering redundant degrees of freedom" [21, p. 127]. In other words, equilibrium-point models organize multiple degrees of freedom to reach a goal point, but not necessarily in an efficient way that reduces a quantity such as time, jerk, or energy.

With the robot weightlifter, one possible implementation of equilibrium-point control involves the use of a single movement primitive that brings the manipulator to the goal configuration. Indeed, with no payload and with no obstacle a simple linear feedback controller accomplishes the task, although this solution fails with payloads as small as 0.5 kg. Instead, I add a second movement primitive for the via point shown in Figure 3.1b. This via point represents the knowledge that certain configurations avoid the leftmost obstacle and also reduce the effective moment arm for the proximal joint. I assume that the via point is obtained through imitation of a successful lift or through instruction from a coach—a human programmer in this case. *The via point is intended to convey crude path information, with no detailed knowledge of any successful trajectory.*

Convergence first to the via point and then to the goal, extends the payload capacity to about 2 kg, beyond which adaptation is necessary. Rather than turn to relatively complicated virtual trajectories as a way to exploit dynamics, I instead use a small number of movement primitives that act as the starting point for learning. The result is a hierarchical motor program that runs three feedback controllers: one for the via point and one for the goal, both adjustable, followed by another, fixed controller to regulate locally about the goal. This converts the original closed-loop equilibrium-point solution, with no via points, to a hierarchical control scheme, i.e., to a "ballpark" open-loop solution [65, 124] that handles minor errors at a lower, closed-loop level of control.

3.4.1 Implementation

To build the initial motor program, I first construct two proportional-derivative (PD) controllers, PD_1 and PD_2:

$$\text{PD}_i: \quad \boldsymbol{\tau}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \mathbf{W}_i[K_p(\boldsymbol{\theta}_i^* - \boldsymbol{\theta}) - K_d\dot{\boldsymbol{\theta}}], \tag{3.1}$$

where $\boldsymbol{\theta} \in \Re^3$ and $\dot{\boldsymbol{\theta}} \in \Re^3$ are the joint positions and velocities, respectively, and $\boldsymbol{\tau} \in \Re^3$ is a vector of joint torques subject to saturation at the ± 50 Nm torque limit. In Eq. (3.1) $\mathbf{W}_i$ is a 3x3 gain matrix, $\boldsymbol{\theta}_i^* \in \Re^3$ is the target equilibrium point, and $K_p = 1000$ Nm·rad^{-1} and $K_d = 250$ Nm·s·rad^{-1} are the nominal proportional and derivative gains, respectively. The target equilibrium points, $\boldsymbol{\theta}_1^*$ and $\boldsymbol{\theta}_2^*$, are initialized to the via point and goal configuration, respectively. Both $\mathbf{W}_1$ and $\mathbf{W}_2$ are initialized to the identity matrix, and so each PD controller initially acts as three uncoupled, scalar-output servomechanisms (one for each joint). Although Eq. (3.1) describes a linear controller, saturation of the output represents a set of nonlinear constraints, possibly inactive, that the learning system can exploit.

Next, the robot executes an "imitation" trial by running to convergence PD_1 followed by PD_2. (A threshold test on the position error establishes convergence.) At convergence the learning system records t_1^* and t_2^*—the time elapsed for the corresponding controller since the start of the trial. Together t_1^* and t_2^* mark the switch times for the open-loop level of control. In particular, the program runs PD_1 from the start of each new trial (time $t = 0$) until $t = t_1^*$, then a switch is made to PD_2 which runs until $t = t_2^*$, followed by the third controller, PD_3, that runs until the trial terminates. PD_3 is a fixed copy of PD_2 that increases the flexibility of the learning system while providing convergence guarantees for movements close enough to the goal. However, PD_3 plays no role during the imitation trial.

The open-loop level of the motor program has two free parameters, the switch times, that are adjusted by trial-and-error learning (described shortly). Together, PD_1 and PD_2 have 24 free parameters, six from $\boldsymbol{\theta}_i^*$ and 18 from $\mathbf{W}_i$, that are adapted with the same learning algorithm. As a form of shaping, I increase the payload at a rate of 0.25 kg every 250 trials of learning. And to speed up learning, which is confounded by interactions between levels, with each new payload the algorithm runs in two phases: a shorter phase

(50 trials) that adjusts the open-loop timing of the motor program, followed by a longer phase (200 trials) that adjusts the lower-level PD controllers while holding the t_i^* fixed.

3.4.2 Direct Policy Search

Table 3.1 summarizes the *simple random search* (SRS) algorithm developed for the robot weightlifting task. The algorithm—a generalization of one described by Luus and Jaakola [95]—performs random search in a K-dimensional parameter space, centered at a base point, $\mathbf{x}$. For the motor program described above, $K = 2$ at the open-loop level of control and $K = 24$ at the closed-loop level. Perturbations, $\Delta\mathbf{x}$, are normally distributed with zero mean and standard deviation equal to the search size, σ. Each test point $(\mathbf{x} + \Delta\mathbf{x})$ is evaluated and the best observed point is kept as the algorithm's return value. For the weightlifting task, the EVALUATE subroutine returns the total effort during a trial with the new parameter settings. Updates to the base point are made by taking a step toward the most recent test point with probability β or else toward the best observed point with probability $1 - \beta$. Thus, β provides one means of control over the degree of exploration, with the extremes of a pure random walk ($\beta = 1$) and of movement along a rough estimate of the gradient ($\beta = 0$). Even with β set to zero, considerable exploration is possible for large values of σ, which decays by a factor γ after each iteration, to the minimum σ_{min}.

The SRS algorithm has several properties that make it a nice choice for trial-and-error learning. First, SRS is easy to implement for a wide variety of optimization problems. Like other direct search methods [141], the SRS algorithm needs no derivative information—an important feature when the cost function is non-differentiable or when the gradient is difficult to estimate (as with deterministic policies subject to noise). The algorithm also has some neurobiological support based on observed random synaptic variation [6]. And compared to "pattern search" algorithms [24], such as the simplex method, SRS makes rapid progress in high-dimensional spaces where some algorithms first require numerous exploratory moves to establish a search direction or to build a simplex, for instance.

Table 3.1. The simple random search (SRS) algorithm.

input
  initial point $\mathbf{x} \in \Re^K$
  step size $\alpha \in [0,1]$
  search strategy $\beta \in [0,1]$
  search size $\sigma \geq 0$
  search decay factor $\gamma \in [0,1]$
  minimum search size $\sigma_{min} \geq 0$
initialize
  $\mathbf{x}_{best} \leftarrow \mathbf{x}$
  $y_{best} \leftarrow \texttt{EVALUATE}(\mathbf{x}_{best})$
repeat
  1. $\Delta\mathbf{x} \leftarrow \mathbf{N}(\mathbf{0}, \sigma)$
  2. $y \leftarrow \texttt{EVALUATE}(\mathbf{x} + \Delta\mathbf{x})$
  3. **if** $y < y_{best}$
  4. $\mathbf{x}_{best} \leftarrow \mathbf{x} + \Delta\mathbf{x}$
  5. $y_{best} \leftarrow y$
  6. $\mathbf{x} \leftarrow \begin{cases} \mathbf{x} + \alpha \cdot \Delta\mathbf{x}, & \text{with prob. } \beta \\ \mathbf{x} + \alpha[\mathbf{x}_{best} - \mathbf{x}], & \text{with prob. } 1 - \beta \end{cases}$
  7. $\sigma \leftarrow \texttt{max}(\gamma\sigma, \sigma_{min})$
until convergence or number of iterations too large
return $\mathbf{x}_{best}$

For the results described shortly, the SRS algorithm step size and search strategy parameters were set to $\alpha = 0.3$ and to $\beta = 0$, respectively. At the motor program level, the initial search size was $\sigma = 0.10$ with a minimum of $\sigma_{min} = 0.01$ and a decay factor of $\gamma = 0.95$. For the PD controllers, the corresponding parameter values were $\sigma = 0.05$, $\sigma_{min} = 0.01$, and $\gamma = 0.99$.

3.5 Results

With no payload, the robot weightlifter could select from a number of qualitatively different solutions to reach the goal configuration. For example, the simple equilibrium-point solution *with no obstacle* follows a direct path from the start to the goal, whereas the "imitation" trial takes a longer path through configuration space, first converging to the via point. Figures 3.2a and 3.2b show, respectively, the configuration space paths for these

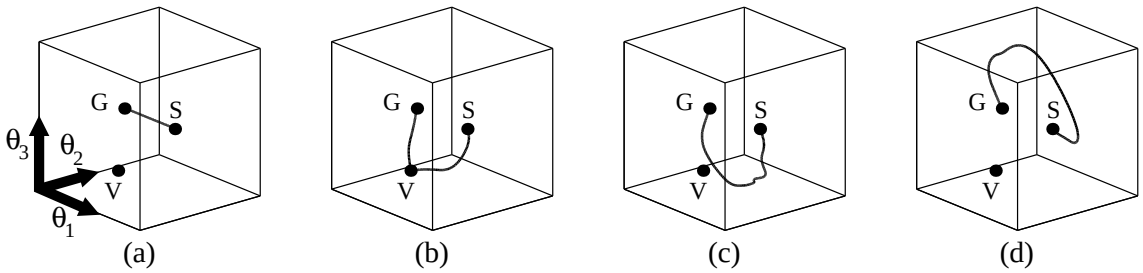

Figure 3.2. Configuration-space trajectories for (a) the simple equilibrium-point solution with no payload, no learning, and no obstacle, (b) the "imitation" trial with no payload, (c) the "standard" solution with a 4.5 kg payload, and (d) the "reversal" solution with a 9.25 kg payload. *S*, *V*, and *G* denote the start, via-point, and goal configurations, respectively.

two solutions which involve no learning. As the payload increases, however, the space of feasible solutions shrinks and adaptation becomes necessary. Essentially, the via point represents an attempt to start the learning system at a favorable place.

The "standard" solution after learning (Figure 3.2c) passes near the via point while coordinating the joints to exploit the robot's intrinsic dynamics. This solution is robust to sensor noise as well as variability in the via point, although with expected degradation in performance as noise increases. During one "lucky" trial—with sources of variability at their maximum—the result of learning was the unexpected "reversal" solution, for which the robot moves through an entirely different region of configuration space. Depicted in Figure 3.2(d), this solution can be made a regular occurrence by simply changing the via point shown previously in Figure 3.1 from -150, -120, and -90 degrees to -315, 135, and 90 degrees (proximal to distal). Figure 3.3 shows a sequence of representative configurations from both the standard and reversal solutions, with the robot near the corresponding via point in frame five of each sequence.

3.5.1 Effects of Learning at the Open-Loop Level

Figure 3.4 shows the evolution of the switch times, t_1^* and t_2^*, for both the standard and the reversal solutions and for 25 learning runs with each of two conditions: one with increasing payload as described above and one with no payload to act as a baseline that

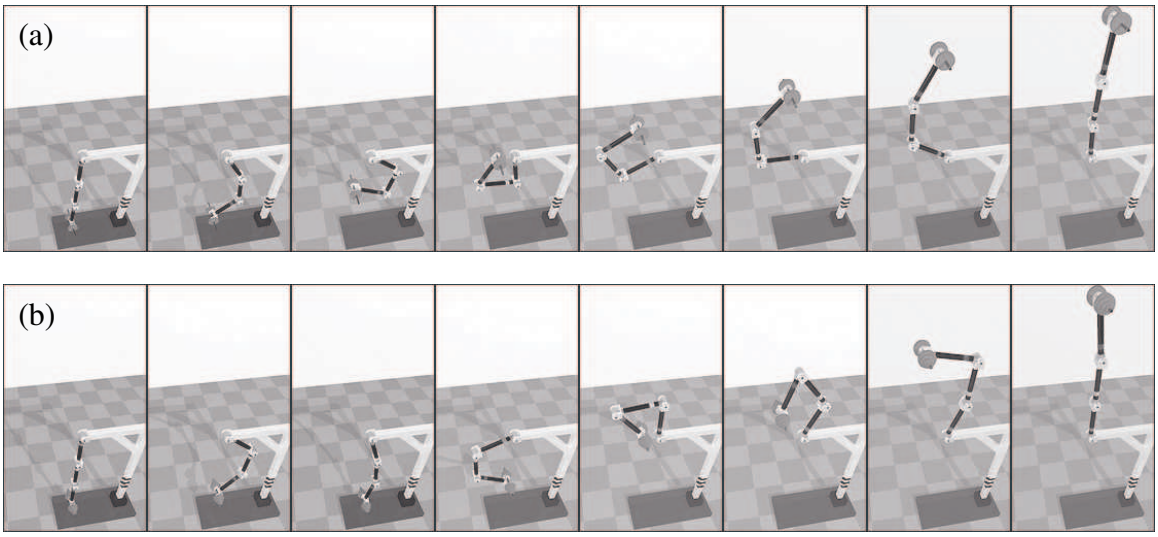

Figure 3.3. Representative examples of (a) the standard solution and (b) the reversal solution.

accounts for the temporal effects of learning. One result not apparent from the figure is that the largest effects of adjusting the switch times occur during the first stage of 50 trials at the very beginning of each learning run. In particular, the system quickly learns that convergence to the via point is unnecessary and that a more efficient solution, in terms of effort, can be had by passing near the via point instead.

After the first block of trials the switch times change very little for the reversal solution and so any observed adaptation in the robot's behavior is largely attributed to learning at the closed-loop level of control. For the standard solution, the effects of subsequent tuning of the switch times is less clear. The observed increase in both t_1^* and t_2^* seems to reflect the overall increase in the movement execution time. The increased movement time, in turn, is related to the increasing payload and to the relative inefficiency of the standard solution compared to the reversal solution. In particular, with the reversal solution the robot generates large velocities at the beginning of the movement and then exploits momentum to a great degree, whereas the standard solution is more of a "brute force" strategy causing the robot to "struggle" and to slow down as the payload increases toward

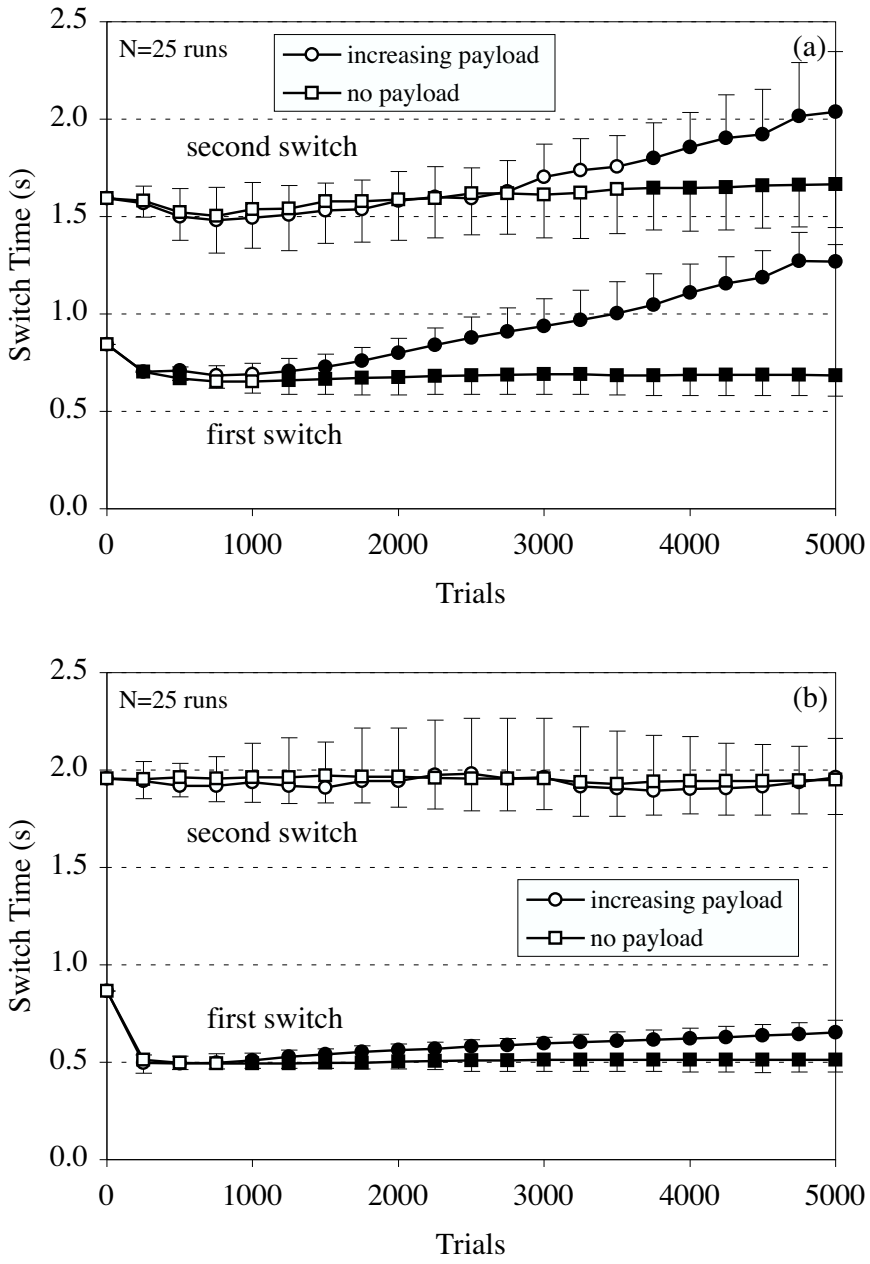

Figure 3.4. Effects of payload and learning on the switch times for (a) the standard solution and (b) the reversal solution. Pairs of solid markers denote statistically significant differences at the corresponding trial number, using a t test with $p < 0.01$. Error bars indicate the standard error of the mean.

the failure point, usually between 4 and 5 kg. A more rigorous analysis, one in terms of dynamic manipulability, is presented in the next chapter.

S24, gain matrix W_1:

$$\begin{bmatrix} 0.70 & 0.21 & 0.08 \\ 0.56 & 1.17 & 0.25 \\ 0.52 & 0.52 & 0.79 \end{bmatrix}$$

S25, gain matrix W_1:

$$\begin{bmatrix} 0.32 & -0.59 & 0.37 \\ 0.37 & 0.90 & 0.31 \\ 0.20 & 0.38 & 0.78 \end{bmatrix}$$

R24, gain matrix W_1:

$$\begin{bmatrix} 1.19 & -0.21 & 0.10 \\ -0.07 & 1.55 & 0.27 \\ 0.65 & 0.49 & 1.17 \end{bmatrix}$$

R25, gain matrix W_1:

$$\begin{bmatrix} 1.15 & -0.34 & 0.15 \\ -0.10 & 0.67 & -0.36 \\ 0.38 & -0.13 & 1.12 \end{bmatrix}$$

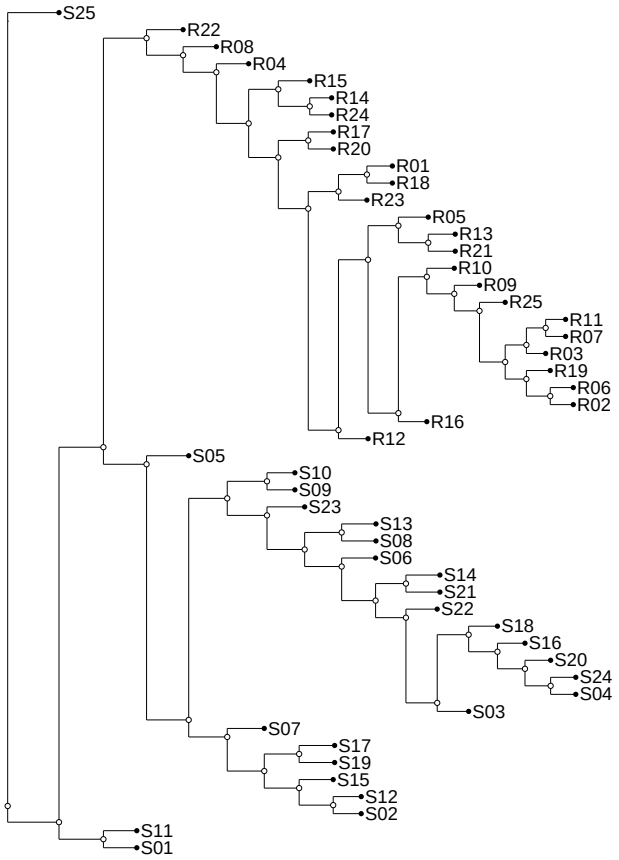

Figure 3.5. Representative gain matrices for the first PD controller and the cluster hierarchy derived from gain matrices for both PD controllers after 2000 trials of learning. Node labels indicate the number of the learning run as well as the solution type: "S" for the standard solution and "R" for the reversal solution.

3.5.2 Effects of Learning at the Closed-Loop Level

At the closed-loop level, most of the adjustments made by the learning algorithm involve the parameters of the gain matrices, and Figure 3.5 shows several representative matrices after 2000 trials of learning. Parameter values were somewhat consistent, with measured standard deviations of about 0.3 for each paramerer. However, there was sufficient variability to prevent general statements about how specific values influence the observed movement of the robot, although one exception comes from learning run "S25" shown in the figure. This learning run was the only observed example where the control system generated an initial swing backward at the proximal joint. For every other example

represented in the figure, the initial torque at the proximal joint was negative, whereas for learning run S25, the initial torque was positive due to the smaller values in the first two entries of the first row of matrix $\mathbf{W}_1$.

Perhaps more informative than the matrices themselves is the cluster hierarchy also shown in Figure 3.5. Each leaf node represents a vector of 18 ordered parameters constructed from the two corresponding matrices, with nine parameters each. The hierarchy was generated by an agglomerative clustering algorithm with Euclidean distance in the associated 18-dimensional parameter space as the metric for building the clusters. Although some variability exists across all learning runs, there is greater consistency for each solution strategy. More specifically, all of the leaf nodes from the reversal solution cluster together in the same sub-tree of the hierarchy, and most of the leaf nodes from the standard solution do the same. Interestingly, run S25 described above is one of the three outliers, again due to the entries in the first row of matrix $\mathbf{W}_1$.

As suggested above, each PD controller initially behaves as three independent servomechanisms—one for each joint—but the learning algorithm transforms them into the robot analogue of a synergy that accounts for intersegmental effects. This happens through the effects of learning on the gain matrices, which have off-diagonal entries that enable active coupling of the individual joints. By changing these entries from their initial values of zero, the learning system essentially causes each joint to become "aware" of the servo errors of its neighbors. To examine the formation of synergies, I quantify active joint coupling as

$$C(\mathrm{PD}_1, \mathrm{PD}_2, \ldots, \mathrm{PD}_M) = 1 - \frac{1}{M} \sum_{i=1}^{M} D(\mathbf{W}_i),$$

where D—the diagonal dominance of an individual gain matrix, $\mathbf{W}$—is defined as

$$D(\mathbf{W}) = \frac{\sum_j |w_{jj}|}{\sum_{j,k} |w_{jk}|},$$

with $M = 2$ for all results presented below.

Figure 3.6 shows the change in C throughout learning—starting with no coupling, i.e., with both $\mathbf{W}_1$ and $\mathbf{W}_2$ diagonal for the initial trial, and increasing to about 0.5 where half the "mass" in the gain matrices falls along the main diagonals. (Not shown in Figure 3.6b is convergence to the 0.5 level after about 7000 trials.) With no payload, the increase in coupling mirrors the drop in effort,[1] and with all but the lightest payloads, the results show a statistically significant increase in coupling over the no-payload condition.

The results in Figure 3.6 demonstrate that active coordination by the control system appears to play an important role for the weightlifting task. However, additional evidence is needed to make a more definitive statement. To begin with, one should always expect the coupling to increase from zero simply because the learning system is not expected to keep the off-diagonal entries of the gain matrices fixed at their initial values (all zero). Thus, the more relevant issue is whether the observed increase in coupling is greater than expected. The no-payload condition provides a useful baseline in this regard, but these results take on greater significance after examining the average total "mass" of the gain matrices, quantified as

$$\frac{1}{M}\sum_{i=1}^{M}\sum_{j,k}|w_{jk}^{i}|,$$

where w_{jk}^{i} denotes an entry of the i^{th} matrix. In particular, Figure 3.7 shows a modest increase in total mass as learning proceeds, from the initial value of 3 (ones along the diagonal) to less than 5.0 after 5000 trials. Even if one assumes that adjustments of the gain matrices occur randomly—with all of the extra mass falling in the off-diagonal entries, on average—then the observed increase in mass is still insufficient to explain all of the observed increase in coupling. In short, the learning system takes advantage of the structure afforded by the gain matrices.

[1] With increasing payload, the effort drops early in the learning process but then increases along with the overall difficulty of the task.

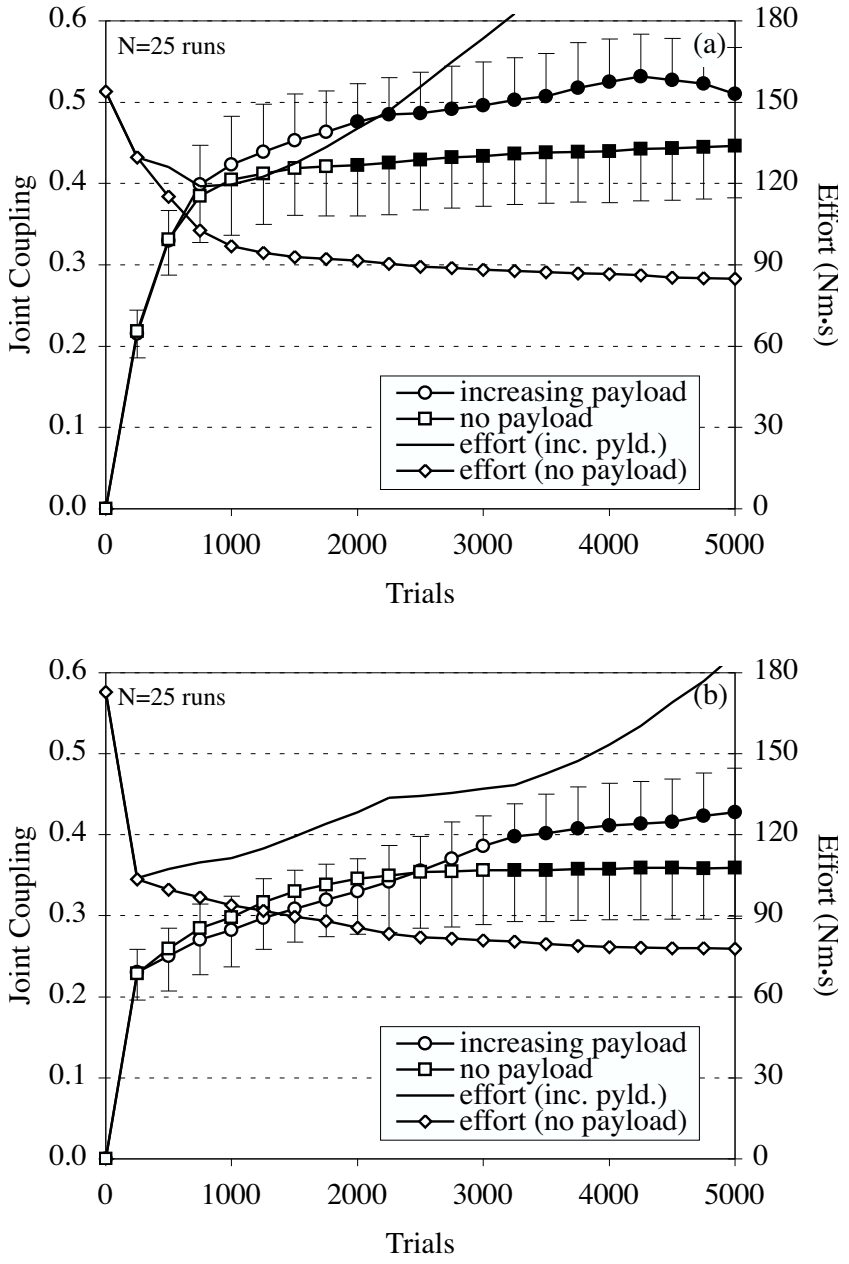

Figure 3.6. Effects of payload and learning on coupling and effort for (a) the standard solution and (b) the reversal solution. Pairs of solid markers denote statistically significant differences ($p < 0.01$) at the corresponding trial number.

The most compelling evidence that active joint coupling carries functional significance for the robot weightlifter comes from a series of experimental manipulations. For one extreme condition the gain matrices were held fixed at their initial values, i.e., at identity, effectively eliminating them from the control system. At the other extreme all entries re-

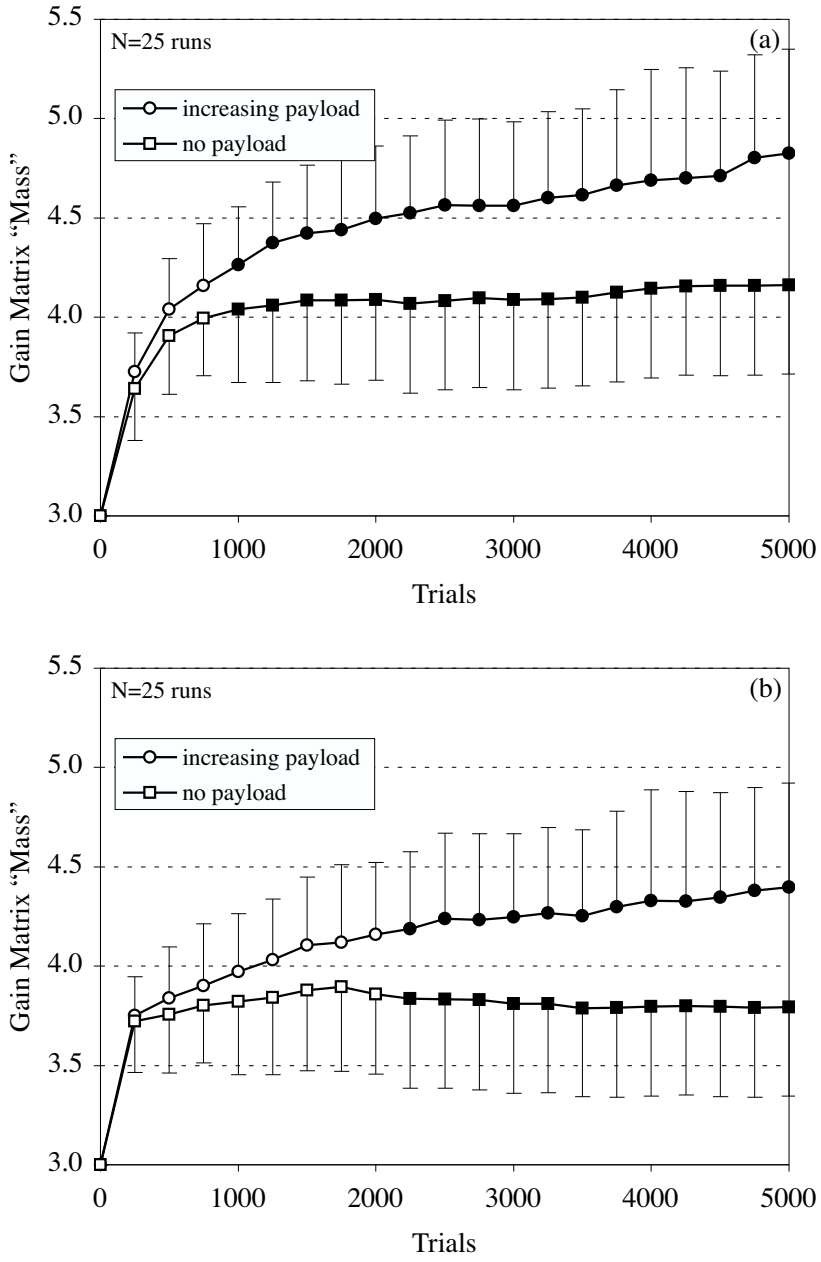

Figure 3.7. Effects of payload and learning on gain matrix "mass" for (a) the standard solution and (b) the reversal solution. Pairs of solid markers denote statistically significant differences ($p < 0.01$) at the corresponding trial number.

mained free parameters for the learning system to adjust as described above. Respectively, these conditions are denoted "Identity" and "Free" in Figures 3.8 and 3.9 below. In between these extremes are several conditions where the entries of the gain matrices are free parameters, but with the added constraint that the coupling never exceed a specified value.

Whenever the SRS algorithm selects a test point that violates this constraint, the learning system draws additional normally distributed samples until an admissible test point is found.

Figure 3.8 shows the effects of coupling on the maximum payload lifted successfully at the end of each learning run. One prominent feature in the figure is the statistically significant advantage that the reversal solution holds over the standard solution. This qualitative result is supported as well by Figure 3.9, which shows the effects of coupling on the effort used to lift a 2 kg payload. However, one caveat is that for both conditions where coupling is eliminated, the robot performs worse with the reversal solution than with the standard solution.

Figures 3.8 and 3.9 actually show results from experimental manipulation of the via points as well as the coupling. In particular, the figures show the effects of coupling for two sets of conditions: one where the via points are free parameters for the learning system, and one where the via points are held fixed at their initial values. There are no statistically significant differences between these sets of conditions; all of the observed differences are due to manipulation of the coupling. Moreover, for both solution strategies these differences show that without active joint coupling the robot is considerably impaired in terms of both efficiency and success at the primary task of lifting a heavy payload. With regard to maximum payload, for instance, the coupling enables more than 75% improvement for the standard solution and more than ten-fold improvement for the reversal solution.

3.5.3 Case Study With a Real Robot

As a "proof of concept" I designed a low-cost robot to demonstrate the successful application of the SRS algorithm with real hardware. Figure 3.10 shows a close-up of the robot near its via point for the standard solution. The joints were actuated by small servomotors, with position commands sent by a host computer at a rate of 10 Hz. Factory-supplied control circuits performed an unknown, nonlinear mapping from position command to motor

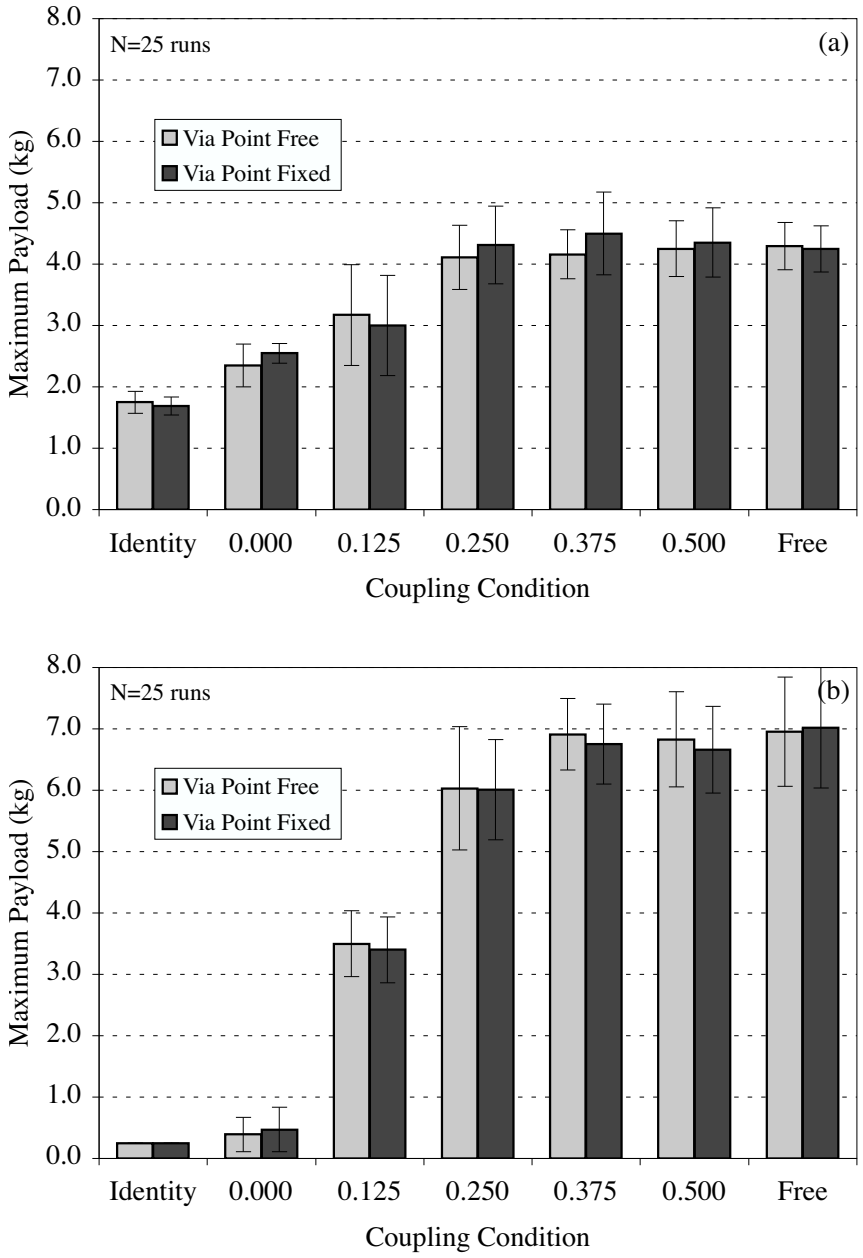

Figure 3.8. Effects of coupling and via points on maximum payload for (a) the standard solution and (b) the reversal solution. The coupling condition refers to the makeup of the gain matrix throughout learning, with numerical labels indicating the maximum allowed coupling.

current, with velocity-dependent adjustments made by additional circuitry that sensed each motor's back-EMF. Control of the robot was complicated by these and other unmodeled effects such as backlash and friction in the motor gearboxes, sensor noise from the feedback

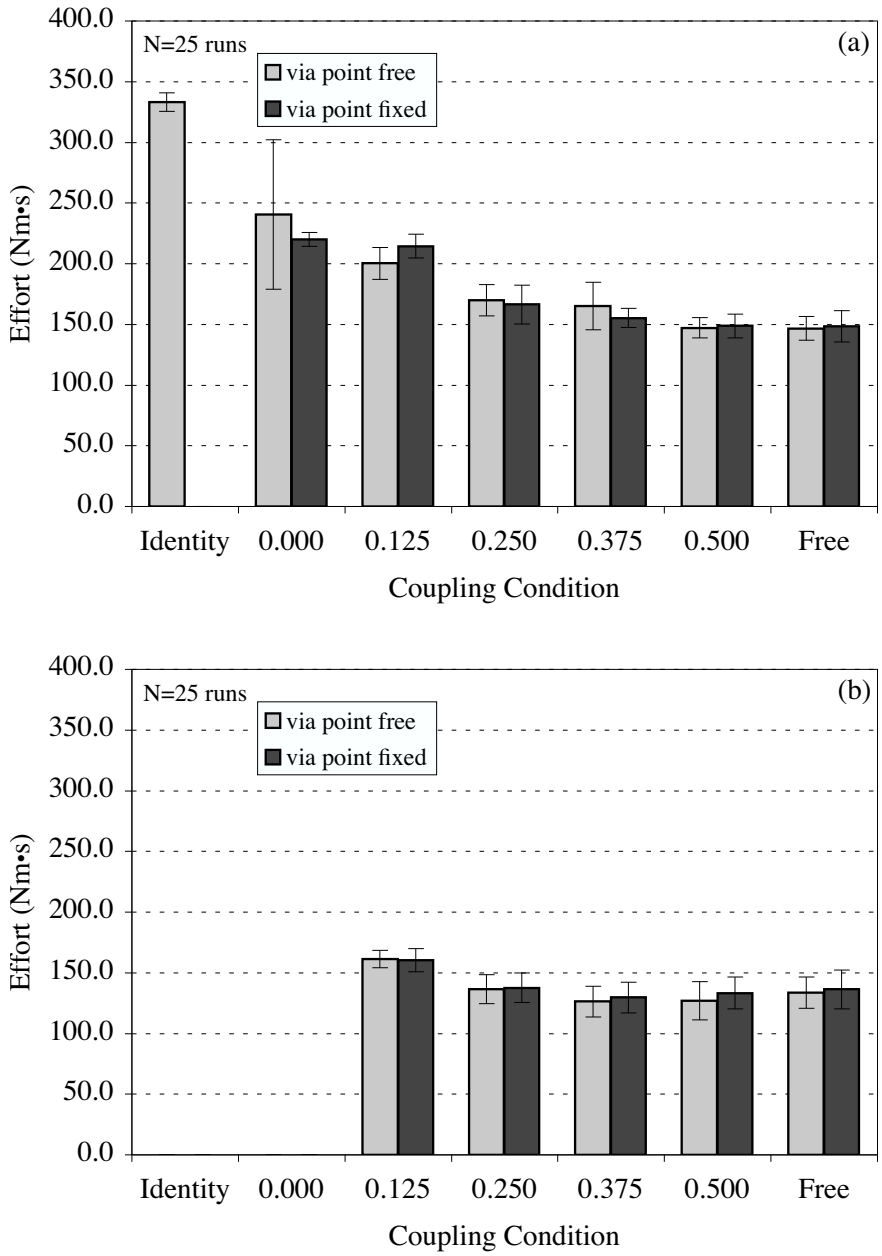

Figure 3.9. Effects of coupling and via points on effort with a 2 kg payload for (a) the standard solution and (b) the reversal solution. The coupling condition refers to the makeup of the gain matrix throughout learning, with numerical labels indicating the maximum allowed coupling. Data apparently missing from the plots correspond to conditions where the robot was unsuccessful for every attempt at the 2 kg level.

potentiometers, and variability in the robot's configuration at the start of each trial. Neither torque nor current feedback was available to quantify the robot's effort, and so the task was modified to use a minimum-time performance criterion.

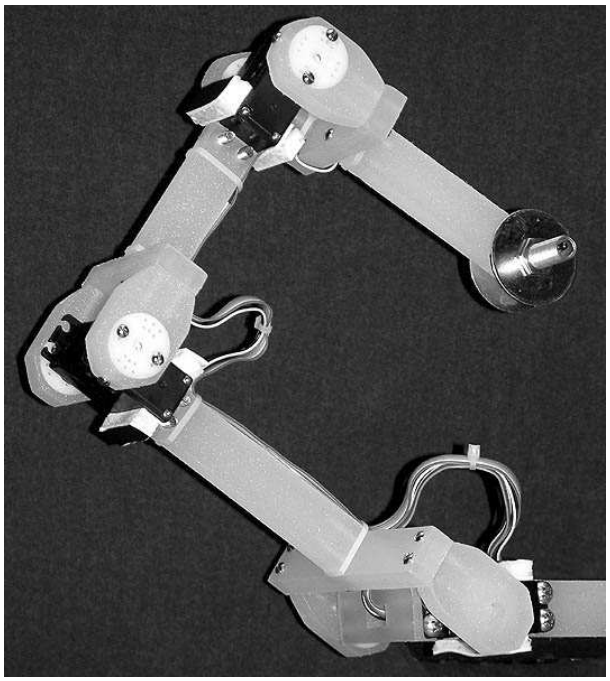

Figure 3.10. Real robot weightlifter near the via point for the standard solution.

Figure 3.11 shows the robot during representative trials using the standard and reversal solutions. Through learning, the real robot approximately doubled its payload capacity, and much like the simulated robot, that capacity was about 50% more for the reversal solution, both before and after learning. For the examples in the figure, active joint coupling increased from zero to about 0.3 after 300 trials.

Extensive experiments that confirm the benefits of joint coupling were infeasible with this robot, although Figure 3.12 illustrates one way that learning improved performance. The leftmost panel shows that without learning, the payload caused the robot to bend downward, away from the via point configuration shown previously in Figure 3.10. In turn, this caused the robot to follow an inefficient path horizontally (panel b) before proceeding to the goal. With learning and the same payload, however, the robot achieved a more favorable configuration prior to the first switch time (panel c). For the remainder of the movement the payload followed a nearly straight path toward the goal as illustrated by the configuration in the rightmost panel. For this particular demonstration, the observed improvements were

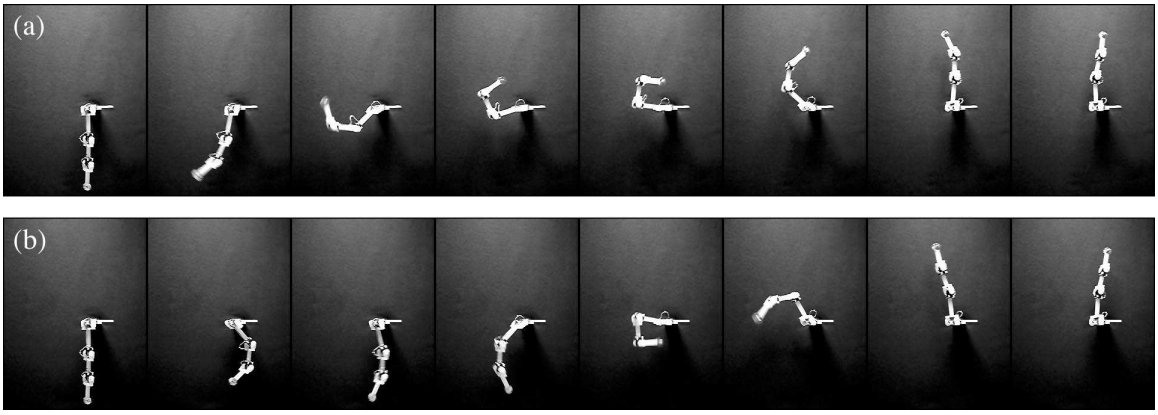

Figure 3.11. Representative examples of (a) the standard solution and (b) the reversal solution for the real robot.

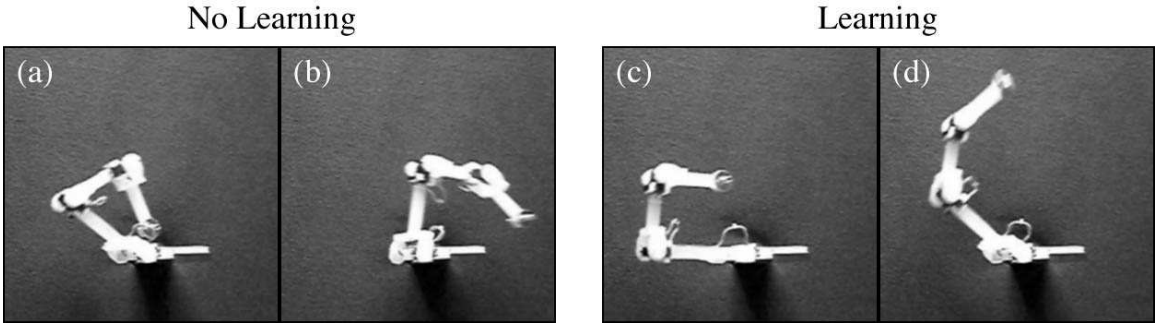

Figure 3.12. Effects of learning for the standard solution during movement toward the via point, (a) and (c), and toward the goal, (b) and (d).

due in part to adjustments of the second component of the via point, i.e., the target angle for the middle joint, although adaptation of the gain matrix appeared to play a larger role.

3.6 Discussion

As suggested in Chapter 1, the research described to this point is largely about motor coordination. However, one machine learning result worth mentioning is that a simple algorithm was able to solve a complicated task—once given the requisite structure in the control policy. More generally, the use of domain knowledge shifts some of the burden when solving a problem onto the designer, or coach, and away from the learning system. General-purpose learning algorithms are certainly desirable although for some types of

problems, such as those involving robot manipulators, strong biases are needed to make machine learning practical. This was especially true for the robot described in the preceding section, since each trial was costly in terms of both experiment time and needed repairs. The approach I demonstrated in this chapter involved the use of a small number of PD controllers for the design of biases that are general for a class of problems, yet easily tailored to the task at hand.

As mentioned in Section 3.4, one criticism of equilibrium-point styles of motor control is the need for virtual trajectories to explain complicated multi-joint movements. Van Ingen Schenau *et al.* [150] also argued that equilibrium-point models are kinematic in nature and, therefore, ill-suited for tasks that exploit dynamics or require control of contact forces. In a similar fashion, Turvey [146] argued not against equilibrium-point models per se, but rather against the chaining of movement primitives as a means for achieving coordination. Although I agree with these criticisms, primitive actions and equilibrium-point control nevertheless play a key role in this thesis. In particular, the movement primitives associated with the via points and the goal configuration supply the hierarchical motor program with an initial kinematic solution that the learning algorithm then transforms into one that exploits dynamics. For the robot weightlifter the observed movement is fluid and whole, with the switch from one PD controller to the next unrecognizable after sufficient learning.

The experimental results for the robot weightlifting task also show that the dominant factor for success is not the timing at the open-loop level, the detailed arrangement of the via points, nor the overall gain of the PD controllers. Instead, the results point to the active joint coupling afforded by the gain matrix as the primary means for coordination. In particular, the observed increase in coupling causes the robot's control system to act as a single functional unit, which I interpret as the formation of a motor synergy.

One way to characterize synergies is that they solve Bernstein's problem by specifying the form of coupling needed to reduce the number of effective degrees of freedom. For instance, locking a joint specifies a rigid coupling between two linkages. A more sophis-

ticated synergy might involve a linear relationship between joint motions, as demonstrated by McDonald *et al.* [99] for dart throwing, by Van Emmerik [148] for drawing movements, and by Vereijken *et al.* [151] for a slalom-like ski task. These studies all concluded that, with practice, participants release degrees of freedom and decouple individual joints, as reflected by a decrease in cross-correlations of the various joint motions. The results in Section 3.5.2 apparently contradict these studies. However, the observed increase in coupling by the gain matrices does not imply any particular relationship among the *external* joint motions—due to the dynamic effects of the manipulator as well is the internal control system.

The main contribution of this chapter was a particular model of motor skill acquisition that allowed us to witness the formation of synergies that exploit dynamics. Moreover, these synergies evolved in a way contrary to the state of the art for industrial robots, whereby feedback linearization is used to design a control system that decouples each joint from the others. The observed increase in coupling demonstrated a progression from independent PD controllers to a "smart" nonlinear controller with substantial communication among components. In conclusion, these observations point me toward the following hypothesis for human coordination: Practice, i.e., learning, transforms a simple, perhaps latent synergy into one that is more sophisticated with tighter coupling from the internal control system, despite any observed decoupling of the kinematic variables.

CHAPTER 4

VELOCITY-DEPENDENT DYNAMIC MANIPULABILITY

The previous chapter demonstrated statistically significant differences between the two basic solutions to the robot weightlifting task. These solutions differ not only in terms of efficiency and maximum payload achieved but also in the form of the movement, i.e., each solution visits a separate region of the robot's configuration space. In this chapter I provide a theoretical analysis which shows that the robot has different capabilities in these regions. This result, in turn, suggests a general strategy for choosing via points for certain tasks, like weightlifting, that require a robot to exploit its velocity-dependent dynamics.

4.1 Manipulability as a Measure of Robot Performance

Any suitable measure that summarizes the capabilities of a manipulator may provide useful information for both the design of multi-purpose robots and for the subsequent planning or learning of efficient movements. Yoshikawa [166] suggested one such measure based on the volume of the *manipulability ellipsoid*, as derived from a manipulator's kinematic properties, i.e., the Jacobian. Similarly, Chiu [41] viewed a manipulator as a "mechanical transformer" and used the Jacobian to describe the duality between velocity and force transmission capabilities. Chiu also defined the *compatibility index* as a basis for computing postures that optimize a robot's performance at a particular task [41]. One canonical example is a planar manipulator with revolute joints and the arm fully "extended" to the edge of its work space. In such configurations, the robot has lost the (instantaneous) ability to move its end-effector inward. The dual, however, is that the manipulator can

withstand any inward force applied at its end-effector, up to the limits of the load-bearing structures at each joint.

A number of variations of this work dealt with dynamics in addition to the kinematic characteristics of a robot. For instance, Asada [9] described the generalized inertia ellipsoid (GIE) that characterizes the effective inertia of a manipulator with reference to forces applied at the end-effector. In other words, the GIE describes how the end-effector will move due to an external load, rather than due to its own actuators. In contrast, Yoshikawa later proposed the *dynamic manipulability measure* [165] which quantifies acceleration capabilities by incorporating the manipulator mass matrix in addition to the Jacobian. Subsequently, Chiacchio and colleagues demonstrated that gravity induces a translation of the *dynamic* manipulability ellipsoid [40] and also that a weighted Jacobian—one that accounts for inertia and torque limits—provides a better match between such ellipsoids and the corresponding acceleration polytopes [39]. (Details follow shortly.)

Despite considerable progress with regard to measures of dynamic manipulability, relatively little attention has been paid to the effects of velocity-dependent dynamics, i.e., Coriolis and centrifugal forces. One exception is the *acceleration radius* [64] which specifies a lower bound on the isotropic acceleration capabilities of the end-effector from any admissible state. Another exception is the *motion isotropy hypersurface* [28] that generalizes the acceleration radius by handling the qualitative mismatch between translational and rotational coordinates, cf. [54].

The drawback of such analyses, however, is their emphasis on isotropic capabilities and worst-case performance. A manipulator may be highly efficient at accelerating its end-effector along some trajectories, even when its isotropic capabilities are diminished or lost altogether. As shown in Section 4.3.3, this scenario is precisely the one observed for the robot weightlifting task of Chapter 3. The goal of this chapter is to provide further insight about the role that velocity plays for manipulability. In particular, I demonstrate

that internal motion of a redundant manipulator as well as movement of its end-effector can have a complex, non-negligible effect on a robot's acceleration performance.

4.2 Dynamic Manipulability

As described in Chapter 2, the equation of motion for an open-chain manipulator with n rigid links and with no friction can be expressed as

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}), \tag{4.1}$$

where $\boldsymbol{\tau}$ is an $n \times 1$ vector of joint actuator torques and $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$ are $n \times 1$ vectors of generalized joint positions, velocities, and accelerations, respectively. In Eq. (4.1), $\mathbf{M}(\mathbf{q})$ is the $n \times n$ mass matrix that captures the configuration-dependent inertial properties of the robot, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ accounts for Coriolis and centrifugal forces, and $\mathbf{G}(\mathbf{q})$, represents the vector of joint torques due to gravity.

Let $\mathbf{x} = [x_1 \ x_2 \cdots \ x_m]^{\mathrm{T}}$ denote the m-dimensional task vector associated with the tip of the manipulator. In this thesis I consider only translational accelerations of the end-effector, not rotational accelerations, and so $\mathbf{x}$ is the Cartesian position of the end-effector with $m \leq 3$. The mapping of positions from joint space to task space is nonlinear, and the $m \times n$ Jacobian matrix $\mathbf{J}$ represents the first-order term in a Taylor expansion of this mapping. Thus the Jacobian also describes the configuration-dependent relationship between velocities in the two coordinate systems:

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}. \tag{4.2}$$

Differentiating Eq. (4.2) with respect to time yields the corresponding relationship for accelerations:

$$\ddot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}. \tag{4.3}$$

4.2.1 Ellipsoid Derivation

Since the manipulator mass matrix is positive definite and, therefore, invertible, one can solve Eq. (4.1) for $\ddot{\mathbf{q}}$ and substitute the result into Eq. (4.3). Dropping the explicit dependencies on $\mathbf{q}$ and $\dot{\mathbf{q}}$, the result is the following expression in terms of actuator torques rather than joint accelerations:

$$\ddot{\mathbf{x}} = \mathbf{J}\mathbf{M}^{-1}(\boldsymbol{\tau} - \mathbf{C} - \mathbf{G}) + \dot{\mathbf{J}}\dot{\mathbf{q}} \tag{4.4}$$

$$= \mathbf{J}\mathbf{M}^{-1}\boldsymbol{\tau} + \ddot{\mathbf{x}}_{vel} + \ddot{\mathbf{x}}_{grav}, \tag{4.5}$$

where

$$\ddot{\mathbf{x}}_{vel} = -\mathbf{J}\mathbf{M}^{-1}\mathbf{C} + \dot{\mathbf{J}}\dot{\mathbf{q}} \tag{4.6}$$

and

$$\ddot{\mathbf{x}}_{grav} = -\mathbf{J}\mathbf{M}^{-1}\mathbf{G}. \tag{4.7}$$

As in [39], if we assume symmetric torque limits such that

$$-\tau_i^{limit} \leq \tau_i \leq +\tau_i^{limit}, \qquad i = 1, \ldots, n,$$

then the normalized actuator torques, $\tilde{\boldsymbol{\tau}}$, can be written as

$$\tilde{\boldsymbol{\tau}} = \mathbf{L}^{-1}\boldsymbol{\tau},$$

where $\mathbf{L} = \text{diag}(\tau_1^{limit}, \ldots, \tau_n^{limit})$. The set of admissible torques can then be represented as a unit hypercube defined by $2n$ inequalities written in the following compact form [39]:

$$\|\tilde{\boldsymbol{\tau}}\|_\infty \leq 1.$$

Substituting $\mathbf{L}\tilde{\boldsymbol{\tau}}$ for $\boldsymbol{\tau}$ in Eq. (4.5) yields

$$\ddot{\mathbf{x}} = \mathbf{J}\mathbf{M}^{-1}\mathbf{L}\tilde{\boldsymbol{\tau}} + \ddot{\mathbf{x}}_{vel} + \ddot{\mathbf{x}}_{grav} \tag{4.8}$$

$$= \mathbf{J}\mathbf{M}^{-1}\mathbf{L}\tilde{\boldsymbol{\tau}} + \ddot{\mathbf{x}}_{bias}, \tag{4.9}$$

where $\ddot{\mathbf{x}}_{bias} = \ddot{\mathbf{x}}_{vel} + \ddot{\mathbf{x}}_{grav}$ is a bias term that represents the end-effector acceleration when the joint actuators are quiescent, i.e., when $\tilde{\boldsymbol{\tau}} = \mathbf{0}$.

Eq. (4.9) maps the *n*-dimensional hypercube defined by $\|\tilde{\boldsymbol{\tau}}\|_\infty \leq 1$ to a polytope of dimension *m* that delimits the set of feasible end-effector accelerations. Alternatively, Eq. (4.9) can be used to map the *n*-dimensional sphere defined by

$$\tilde{\boldsymbol{\tau}}^{\mathrm{T}}\tilde{\boldsymbol{\tau}} \leq 1 \tag{4.10}$$

to an *m*-dimensional ellipsoid. This *dynamic manipulability ellipsoid* is derived by solving Eq. (4.9) for $\tilde{\boldsymbol{\tau}}$ and by substituting the result into Eq. (4.10):

$$[(\mathbf{J}\mathbf{M}^{-1}\mathbf{L})^{-1}(\ddot{\mathbf{x}} - \ddot{\mathbf{x}}_{bias})]^{\mathrm{T}}[(\mathbf{J}\mathbf{M}^{-1}\mathbf{L})^{-1}(\ddot{\mathbf{x}} - \ddot{\mathbf{x}}_{bias})] \leq 1 \tag{4.11}$$

$$(\ddot{\mathbf{x}} - \ddot{\mathbf{x}}_{bias})^{\mathrm{T}}(\mathbf{L}^{-1}\mathbf{M}\mathbf{J}^{-1})^{\mathrm{T}}(\mathbf{L}^{-1}\mathbf{M}\mathbf{J}^{-1})(\ddot{\mathbf{x}} - \ddot{\mathbf{x}}_{bias}) \leq 1 \tag{4.12}$$

$$(\ddot{\mathbf{x}} - \ddot{\mathbf{x}}_{bias})^{\mathrm{T}}(\mathbf{J}^{-1})^{\mathrm{T}}\mathbf{M}^{\mathrm{T}}(\mathbf{L}^{-1})^{\mathrm{T}}\mathbf{L}^{-1}\mathbf{M}\mathbf{J}^{-1}(\ddot{\mathbf{x}} - \ddot{\mathbf{x}}_{bias}) \leq 1. \tag{4.13}$$

Since $\mathbf{M}$ and $\mathbf{L}$ are both symmetric, the previous result simplifies to

$$(\ddot{\mathbf{x}} - \ddot{\mathbf{x}}_{bias})^{\mathrm{T}}(\mathbf{J}^{-\mathrm{T}}\mathbf{M}\mathbf{L}^{-2}\mathbf{M}\mathbf{J}^{-1})(\ddot{\mathbf{x}} - \ddot{\mathbf{x}}_{bias}) \leq 1. \tag{4.14}$$

The matrix $\mathbf{J}^{-\mathbf{T}}\mathbf{M}\mathbf{L}^{-2}\mathbf{M}\mathbf{J}^{-1}$ from Eq. (4.14) determines the shape of the dynamic manipulability ellipsoid. Each eigenvector, $\mathbf{v}_i$, of this matrix specifies one of the ellipsoid's principal axes, the length of which is given by $1/\sqrt{w_i}$, where w_i is the corresponding eigenvalue. The shape of the "kinematic" manipulability ellipsoid, on the other hand, is determined by $\mathbf{J}^{-\mathbf{T}}\mathbf{J}^{-1}$, with no correction for the manipulator's inertia and actuator torque limits.

Computation of the actual ellipsoid requires a suitable inverse for the Jacobian, which, in general, is not square. Following the recommendation by Chiacchio [39], I utilize the weighted pseudoinverse of the Jacobian:

$$\mathbf{J}_Q^{\dagger} = \mathbf{Q}^{-1}\mathbf{J}^{\mathbf{T}}(\mathbf{J}\mathbf{Q}^{-1}\mathbf{J}^{\mathrm{T}})^{-1},$$

where $\mathbf{Q} = \mathbf{M}\mathbf{L}^{-2}\mathbf{M}$ is a weight matrix that accounts for both inertia and torque limits. In summary, the shape of the "weighted" dynamic manipulability ellipsoid is determined by the eigenvectors and eigenvalues of

$$\mathbf{N} = \mathbf{J}_Q^{\dagger^{\mathrm{T}}}\mathbf{Q}\mathbf{J}_Q^{\dagger}.$$

See [39] for further details.

4.2.2 Velocity Effects

Chiacchio *et al.* [40] demonstrated previously that $\ddot{\mathbf{x}}_{grav}$ has the effect of translating the center of the dynamic manipulability ellipsoid away from the origin where $\ddot{\mathbf{x}} = \mathbf{0}$. Even small translations can have a dramatic effect on the achievable accelerations in some directions, especially when $\ddot{\mathbf{x}}_{grav}$ is aligned approximately with the ellipsoid's minor axis. Eqs. (4.8)-(4.14) show that the velocity dependent terms from Eq. (4.6) have a similar effect on dynamic manipulability. Moreover, the overall displacement given by $\ddot{\mathbf{x}}_{bias}$ is sometimes dominated by $\ddot{\mathbf{x}}_{vel}$ as demonstrated in the following section.

Returning for the moment to Eq. (4.1), one can show that $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is bounded by a quadratic in $\dot{\mathbf{q}}$ such that

$$\|\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\| \leq c(\mathbf{q})\|\dot{\mathbf{q}}\|^2,$$

where $c(\mathbf{q})$ is a known scalar function specified by the manipulator's inertial properties and $\|\cdot\|$ is any appropriate norm [89]. Moreover, if the joints are all revolute, then $c(\mathbf{q})$ becomes

a constant independent of configuration [89]. And since $\mathbf{M}^{-1}$ is bounded [46], one can derive similar relationships for both terms in Eq. (4.6) and, therefore, for $\ddot{\mathbf{x}}_{vel}$ as well:

$$\|\ddot{\mathbf{x}}_{vel}(\mathbf{q}, \dot{\mathbf{q}})\| \leq \beta(\mathbf{q})\|\dot{\mathbf{q}}\|^2, \tag{4.15}$$

where $\beta(\mathbf{q})$ is another known scalar function of $\mathbf{q}$.

Eq. (4.15) provides further intuition about the effects of velocity on manipulability. Nominally, the acceleration bias is quadratic in the joint velocity. However, Eq. (4.15) represents an upper bound and provides no indication for which states $\|\ddot{\mathbf{x}}_{vel}\|$ is small even when $\|\dot{\mathbf{q}}\|^2$ is relatively large. Eq. (4.15) also fails to capture the orientation of $\ddot{\mathbf{x}}_{vel}$ which can change dramatically over a short period of time. Theoretical limitations such as these motivate the use of case studies to gain further insight about the role that velocity plays for manipulability.

4.3 Case Studies

The examples studied in this section all utilize the planar three-link manipulator described by Chiacchio [39]. Each link is modeled as a rigid rod with uniform density and with inertial parameters set as follows:

link	length (m)	mass (kg)
1	1.0	4.0
2	0.8	2.0
3	0.5	0.6

Eq. (4.1) summarizes the dynamics of this manipulator, with gravity acting downward and with actuator torques normalized by $\mathbf{L} = \text{diag}(100, 30, 4)$ Nm.

4.3.1 Increasing Velocity

The first case study replicates one by Chiacchio [39] with the addition of a velocity component. In particular, consider the configuration

$$\mathbf{q} = \begin{bmatrix} +\frac{\pi}{2} & -\frac{\pi}{2} & -\frac{\pi}{2} \end{bmatrix}^{\mathrm{T}}$$

shown in Figure 4.1, but with joint velocities $\dot{q}_1 = \dot{q}_2 = \dot{q}_3 = \omega$, for ω increasing from 0 to 4 rad $\cdot$ s^{-1}. For the case $\omega = 1$ rad $\cdot$ s^{-1} we have

$$\mathbf{M} = \begin{bmatrix} 4.494 & 0.711 & -0.100 \\ 0.711 & 0.861 & 0.050 \\ -0.100 & 0.050 & 0.050 \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} 4.440 & -0.680 & -0.480 \end{bmatrix}^{\mathrm{T}},$$

$$\mathbf{G} = \begin{bmatrix} 12.569 & 12.570 & 0.000 \end{bmatrix}^{\mathrm{T}},$$

$$\mathbf{J} = \begin{bmatrix} -0.500 & 0.500 & 0.500 \\ 0.800 & 0.800 & 0.000 \end{bmatrix},$$

and

$$\dot{\mathbf{J}}\dot{\mathbf{q}} = \begin{bmatrix} -3.200 & 3.500 \end{bmatrix}^{\mathrm{T}}.$$

Then from Eqs. (4.6) and (4.7) we have

$$\ddot{\mathbf{x}}_{vel} = \begin{bmatrix} 1.072 & 3.697 \end{bmatrix}^{\mathrm{T}}$$

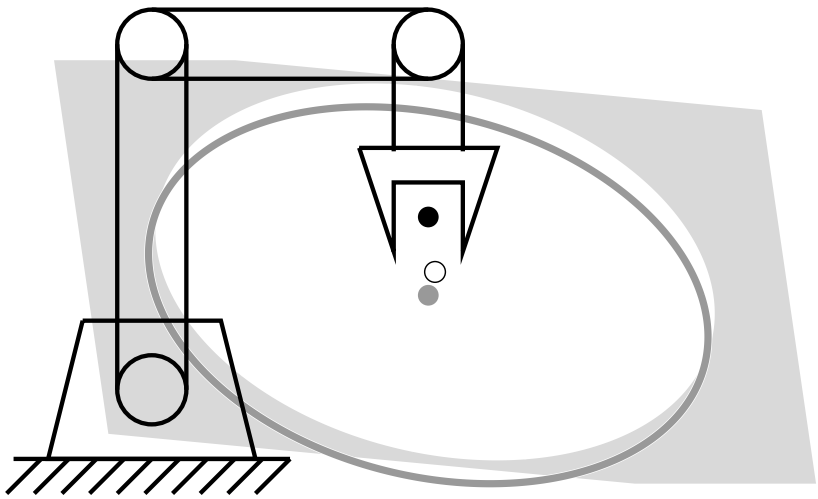

Figure 4.1. Acceleration polytope and dynamic manipulability ellipsoids for a three-link, planar manipulator with $\mathbf{q} = [\frac{+\pi}{2} \ \frac{-\pi}{2} \ \frac{-\pi}{2}]^{\mathrm{T}}$ and $\dot{\mathbf{q}} = [1 \ 1 \ 1]^{\mathrm{T}}$.

and

$$\ddot{\mathbf{x}}_{grav} = \begin{bmatrix} 0.000 & -12.400 \end{bmatrix}^{\mathrm{T}}.$$

Thus for $\omega = 1$, $\|\ddot{\mathbf{x}}_{grav}\|$ is more than three times greater than $\|\ddot{\mathbf{x}}_{vel}\|$. In this case, the overall translation given by $\ddot{\mathbf{x}}_{bias}$ is attributed primarily to the gravity component, as exhibited by a small offset between the ellipsoids in Figure 4.1. In the figure, the polytope (light gray) and corresponding ellipsoid (white cutout) depict the manipulator's true acceleration capabilities with corrections for both gravity and velocity, i.e., with $\ddot{\mathbf{x}}_{bias} = \ddot{\mathbf{x}}_{vel} + \ddot{\mathbf{x}}_{grav}$. Superimposed is the dynamic manipulability ellipsoid (dark gray) with correction for gravity only, i.e., with $\ddot{\mathbf{x}}_{bias} = \ddot{\mathbf{x}}_{grav}$. Small circles denote the position of the end-effector (black), the centroid of the polytope (white), and the center of the gravity-corrected ellipsoid (gray).

Despite the greater influence of the gravity bias in Figure 4.1, Figure 4.2 shows that $\ddot{\mathbf{x}}_{vel}$ grows quadratically in $\|\dot{\mathbf{q}}\|$ as expected. (Empirically, Eq. (4.15) holds with equality for $\beta = 1.28$.) In this particular example, the inner product of $\ddot{\mathbf{x}}_{vel}$ and $\ddot{\mathbf{x}}_{grav}$ is negative, and so $\ddot{\mathbf{x}}_{vel}$ cancels part the offset due to gravity. Shown in Figure 4.2 are the acceleration radius

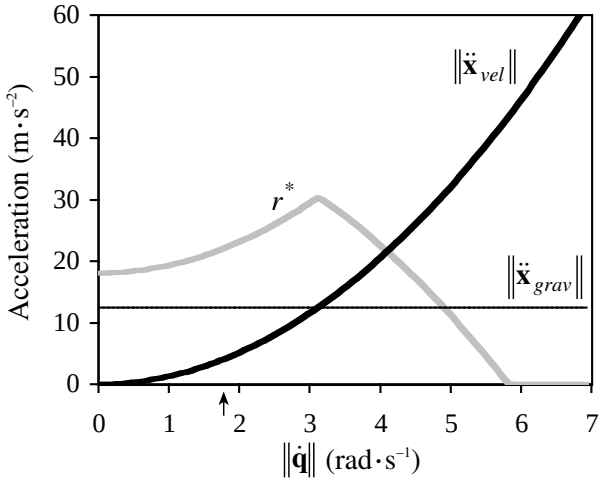

Figure 4.2. Acceleration versus joint velocity magnitude with $\dot{q}_1 = \dot{q}_2 = \dot{q}_3 = \omega$.

(gray curve) and the magnitude of the acceleration offsets due to gravity and velocity (light and heavy curves, respectively). The small arrow marks the value of $\|\dot{\mathbf{q}}\|$ that corresponds to $\omega = 1$ as in Figure 4.1. For $\|\dot{\mathbf{q}}\| < 3$, the effect is improved isotropic capabilities, as indicated by the increased acceleration radius, r^*.[1] When velocity grows beyond about $\|\dot{\mathbf{q}}\| = 3$, however, r^* decreases toward zero, since $\ddot{\mathbf{x}}_{vel}$ grows large enough to dominate $\ddot{\mathbf{x}}_{grav}$ and eventually the polytope no longer encompasses the end-effector.

4.3.2 Exploiting Redundancy

It is well known that redundancy can complicate a control problem but, at the same time, can expand the capabilities of a robot, e.g., for singularity avoidance. One approach for dealing with redundancy is to use heuristic strategies derived from theoretical considerations. For instance, Chiacchio [38] suggested that for minimum-time control, one could exploit redundancy to align more closely the surface of the acceleration ellipsoid with the tangent to the task space path. The next example demonstrates that redundancy can also be

[1] As defined in [64], the acceleration radius represents the size of the largest sphere which is centered on the end-effector and fits within every acceleration polytope over the entire operating range of the manipulator. In this chapter, I relax the latter requirement and plot r^* separately for each state.

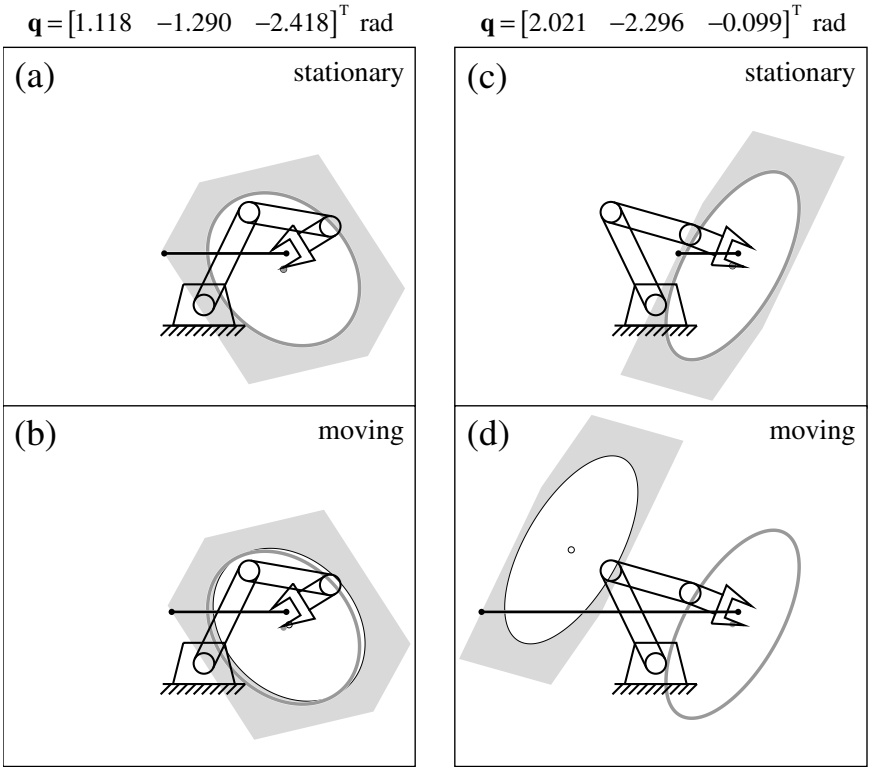

Figure 4.3. Acceleration polytope and dynamic manipulability ellipsoids for extreme postures that maintain the end-effector position at $(0.8, 0.5)$.

used to reconfigure a robot such that internal motion of the manipulator, with no "external" motion of the end-effector, improves the acceleration capabilities along some prespecified direction.

Consider the manipulator shown previously in Figure 4.1, and suppose we wish to reconfigure this robot to maximize the achievable leftward acceleration of the end-effector. In this scenario we also require that the end-effector position remain constant until the reconfiguration is (nearly) complete. One possible solution is to search for the best static posture, with the gravity-corrected acceleration polytope as a means for evaluation. Figure 4.3a shows the outcome of this search, and the small offset between ellipsoids in Figure 4.3b illustrates that velocity (for movement as described shortly) has little effect at the resulting configuration.

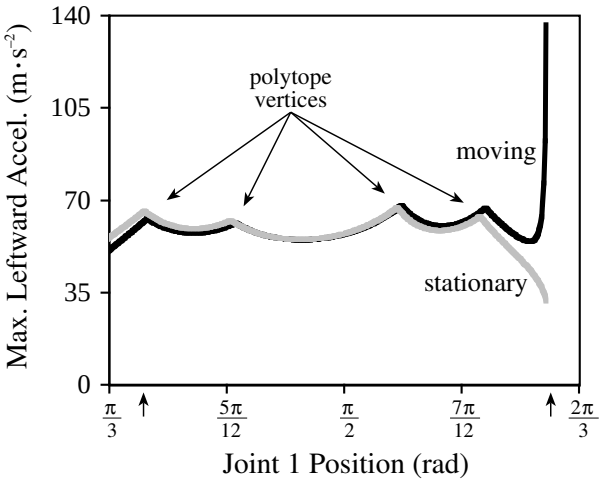

Figure 4.4. Maximum leftward acceleration achievable at each position of joint 1.

However, if we repeat the search while evaluating the polytope corrected for velocity as well as gravity, then the outcome is an entirely different posture with more than twofold improvement in maximum achievable leftward acceleration (depicted by a horizontal line from the end-effector to the leftmost point on the polytope). Shown in Figure 4.3d, this result was generated for the case where joint 1 moves at a constant speed of 1 rad $\cdot$ s^{-1} and the remaining joints move in accordance with the inverse kinematics solution given q_1. Thus, when the manipulator is stationary, the configuration in panel (a) is preferred to that in panel (c). This contrasts with a moving robot, for which the configuration in panel (d) is preferred to the one in panel (b).

As summarized in Figure 4.4, for this example $\ddot{\mathbf{x}}_{vel}$ makes little difference in the maximum leftward acceleration until the manipulator approaches the configuration

$$\mathbf{q} = \begin{bmatrix} 2.02 & -2.34 & 0.00 \end{bmatrix}^{\mathrm{T}},$$

where the two distal-most links become aligned. Shown are the two conditions where the manipulator is either stationary (gray curve) or moving (black curve). Small arrows mark

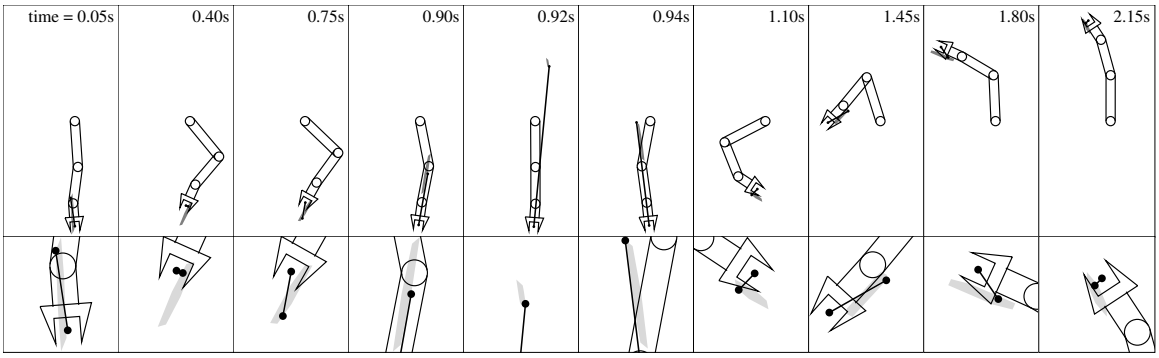

Figure 4.5. Acceleration polytope while raising a 6.5 kg payload to the unstable equilibrium.

the values of q_1 that correspond to the configurations shown Figure 4.3. Notice that local extrema occur whenever the active torque constraint changes at a polytope vertex.

4.3.3 Raising A Payload

The third case study demonstrates the benefits of passing through a singularity when raising a heavy payload. From the previous example we saw that when two links become nearly aligned, motion can lead to large displacements of the acceleration polytope and, therefore, to large discrepancies between analyses that assume the robot is either stationary or not. Large displacements of the polytope can be critical, perhaps catastrophic, to the task at hand. Nevertheless, the example in Figure 4.3d illustrates the potential benefits of large values for $\ddot{\mathbf{x}}_{vel}$. However, that example is unrealistic because the analysis assumed that Cartesian position remains fixed, even when the polytope is displaced enough that translation of the end-effector is unavoidable. This section makes no such assumption and, instead, shows an analysis of a trajectory that emerges from a dynamic simulation of the robot. Figure 4.5 illustrates one such trajectory for the weightlifting task of Chapter 3.

In the figure, the acceleration attained at each time step is depicted as a vector from the end-effector to a point within the acceleration polytope (gray). Perhaps the most prominent feature of Figure 4.5 is the relatively small size of the polytope. (Close-ups are shown in the lower panels.) As expected, the heavy payload has an adverse effect on those capabilities

attributed to the size and shape of the polytope. However, the payload does not necessarily impair the robot's ability to generate a wide range of accelerations at the required moments. This is evidenced by the varied orientation and magnitude of the vectors in the figure.

Of particular interest from this example is motion leading to the singularity near time $t = 0.92$ s (frame five of Figure 4.5). Near the beginning of the movement, at $t = 0.05$ s, all three joint actuators operate at their torque limits and the corresponding end-effector acceleration lies at a vertex of the polytope. Close to time $t = 0.40$ s the robot, with low velocity, achieves a posture that facilitates the next phase of the movement. In particular, at $t = 0.75$ s the manipulator begins a downward acceleration toward the singularity, and by $t = 0.92$ s, a large vertical acceleration "kicks" the payload upward.

As suggested by Figure 4.6a, the upward kick is due almost entirely to the large value of $\ddot{\mathbf{x}}_{vel}$ induced by motion near the singular configuration. The plot shows the acceleration radius (gray curve, barely visible) and the magnitude of the acceleration offsets due to gravity and velocity (light and heavy curves, respectively). The lower plot, Figure 4.6b, shows the potential, kinetic, and total energy during the movement (light, heavy, and gray curves, respectively). Thus, the singularity appears to offer an efficient means for "focusing" kinetic energy to produce movement in a particular direction. The robot then achieves a "folded" configuration, and with a shorter effective moment arm against gravity, the manipulator then appears well suited to generate potential energy. After about $t = 1.5$ s, the remainder of the movement deals primarily with the coordinated transfer of kinetic energy to potential energy, rather than the production of additional energy for the system.

4.4 Posture-Dependent Estimates of the Velocity Bias

The previous example illustrates that the acceleration polytope is helpful for *a posteriori* analysis of an emergent trajectory. However, the present challenge is to incorporate velocity-dependent effects as part of *a priori* analysis of manipulability. Can one still use the acceleration polytope (or ellipsoid) to determine favorable postures with which to bias

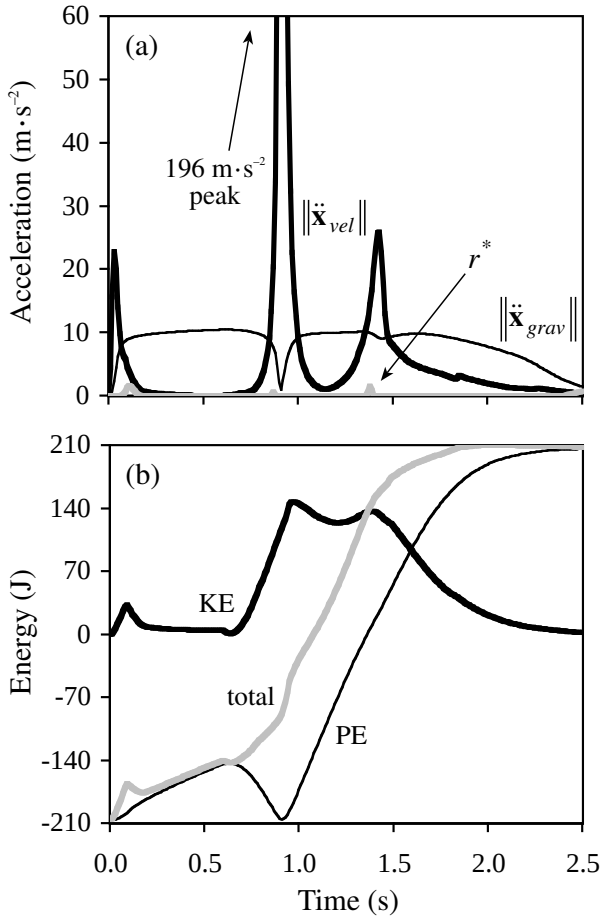

Figure 4.6. Acceleration (a) and energy (b) versus time for the trajectory illustrated in Figure 4.5.

a trajectory planner or learning algorithm? The results to this point suggest that this goal is a difficult undertaking, especially for manipulators that move at high speed. It turns out, however, that one can use Eq. (4.6) to compute an estimate of the velocity bias for a given posture, with no prior knowledge of the actual velocity.

In Eq. (4.6), $\ddot{\mathbf{x}}_{vel}$ depends on velocity explicitly through the term $\dot{\mathbf{J}}\dot{\mathbf{q}}$ and also implicitly through the dependence of both $\mathbf{C}$ and $\dot{\mathbf{J}}$ on $\dot{\mathbf{q}}$. Thus, to compute the true value of $\ddot{\mathbf{x}}_{vel}$ one actually needs the value of $\dot{\mathbf{q}}$, but in lieu of this prior information one can simply choose $\dot{\mathbf{q}}$ randomly according to a distribution consistent with whatever information may be available. Figure 4.7 shows the result of applying Eq. (4.6) for 2500 values of $\dot{\mathbf{q}}$ cho-

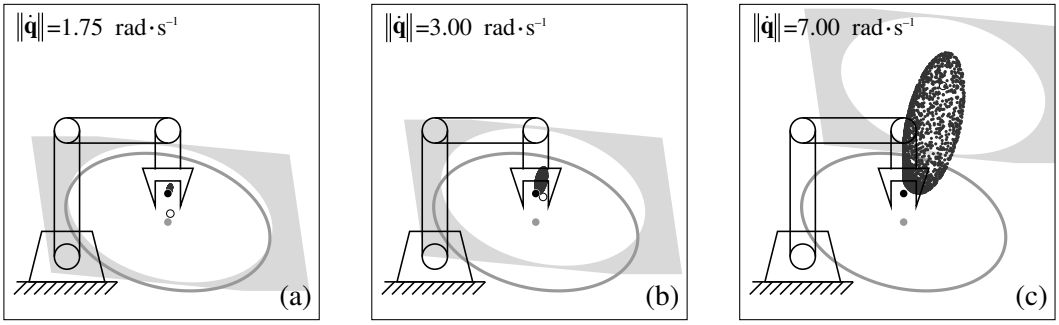

Figure 4.7. Distribution of $\ddot{\mathbf{x}}_{vel}$ for randomly chosen values of $\dot{\mathbf{q}}$.

sen to have constant magnitude yet random orientation. Each cloud of points depicts the distribution of $\ddot{\mathbf{x}}_{vel}$ for the posture shown previously in Figure 4.1. As with the example in Section 4.3.1, the figure also shows the gravity corrected ellipsoid (dark gray) and the acceleration polytope for the case where all three joints move at the same speed.

When $\|\dot{\mathbf{q}}\|$ is small as in Figure 4.7(a), the velocity bias is also small as expected, and the corresponding cloud of points is barely visible. However, panels (b) and (c) show that as velocity grows the distribution of $\ddot{\mathbf{x}}_{vel}$ also grows, yet remains oriented in a direction roughly opposite that of the gravity bias, $\ddot{\mathbf{x}}_{grav}$. This generalizes the qualitative results of Figure 4.2 to situations where the joints may not be moving at the same speed. That is, for increasing magnitude of $\dot{\mathbf{q}}$, the isotropic capabilities of the manipulator first improve for some intermediate values and then diminish until lost altogether when the polytope becomes displaced far enough from the end-effector.

These empirical results demonstrate how one can improve upon the information provided by usual measures of manipulability, which evaluate postures irrespective of movement. However, the approach taken in this section does not preclude the use of approximate knowledge regarding velocity. For example, if the proximal joint is known to be moving slowly, then the randomly distributed values of $\dot{\mathbf{q}}$ can be filtered to discard those values where $|\dot{q}_1|$ is large. Figure 4.8 shows the situation where the (actual or desired) direction of end-effector movement is prespecified; each cloud of points is filtered to include only those values of $\ddot{\mathbf{x}}_{vel}$ for which the orientation of $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$ is within 15 degrees of the arrows

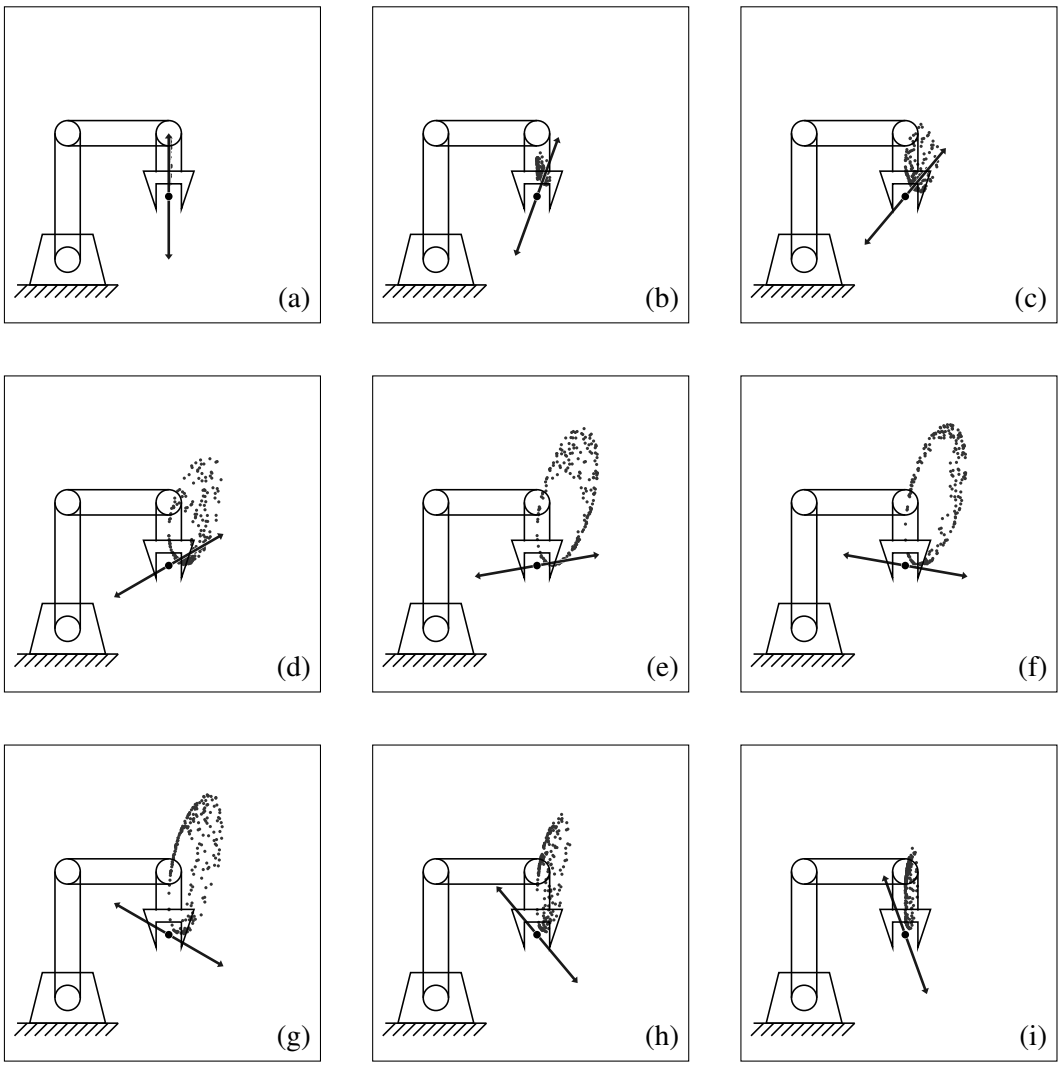

Figure 4.8. Distribution of $\ddot{\mathbf{x}}_{vel}$ for $\|\dot{\mathbf{q}}\| = 7$ and $\dot{\mathbf{q}}$ filtered by the direction of end-effector movement.

shown in each panel.[2] Panel (b), for instance, shows a pair of directions of end-effector movement where the velocity bias is small, despite the large magnitude of $\dot{\mathbf{q}}$. Similarly, panel (e) shows a pair of directions where $\ddot{\mathbf{x}}_{vel}$ is large, though not universally so.

Even with prior knowledge of velocity, the value of $\ddot{\mathbf{x}}_{vel}$ may prove to be of limited use for planning the *details* of an efficient trajectory. More specifically, the velocity bias serves

[2]The velocity bias is quadratic in the joint velocity, and so for arbitrary $\dot{\mathbf{q}}$ the corresponding value of $\ddot{\mathbf{x}}_{vel}$ is the same as that for $-\dot{\mathbf{q}}$. Thus, each "double" arrow in Figure 4.8 is associated with two separate, yet identical distributions of the velocity bias.

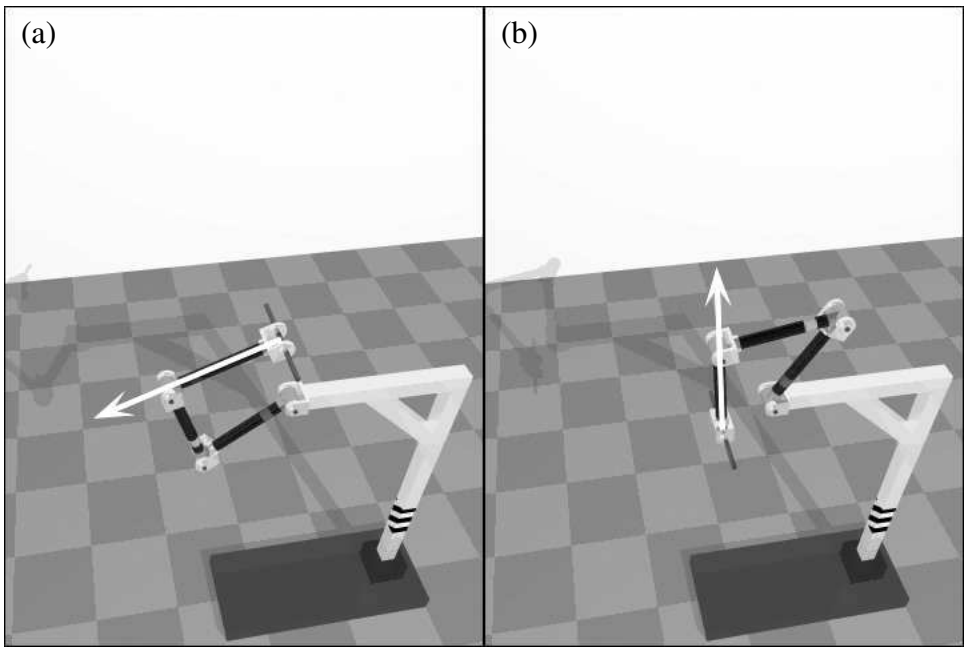

Figure 4.9. Orientation of $\ddot{\mathbf{x}}_{vel}$ for representative postures from (a) the standard and (b) the reversal solution to the weightlifting task.

as an additional performance criterion that complicates an already difficult search procedure based on conventional measures of manipulability. Nevertheless, the results may be extremely beneficial for choosing among qualitatively distinct solutions to a particular task. This kind of "task redundancy" differs from the usual notion of manipulator redundancy, where measures of manipulability are used to optimize movement in the null space of some primary objective, cf. Eq. (2.3). For the weightlifting task of Chapter 3, task redundancy refers to the choice between the "standard" and "reversal" solutions, each of which is also redundant in the usual sense. Figure 4.9 shows representative postures from both solutions, along with arrows that depict the orientation of $\ddot{\mathbf{x}}_{vel}$.[3] For the reversal solution, the velocity bias is directed to facilitate raising of the payload. Regardless of the singularity analysis in Section 4.3.3, the reversal solution therefore appears best suited for the weightlifting task.

[3]The three-link pendulum is a degenerate case such that for randomly chosen joint velocities, $\ddot{\mathbf{x}}_{vel}$ is distributed along a straight line as shown in the figure.

4.5 Discussion

Measures of manipulability are intended as a description of a robot's capabilities—before specification of a task. Implicit in such measures is the assumption that the robot is either motionless (cf. dynamic manipulability measure [166]) or else moving slowly (cf. acceleration radius [64]). Although the analysis of stationary manipulators is "fundamental" [166] for a deeper understanding of the relationship between manipulability ellipsoids and performance, the examples in this chapter demonstrate that motion has a complex, non-negligible effect on dynamic manipulability.

With the weightlifting example in Figure 4.5, for instance, we saw that displacement of the acceleration polytope plays a more instrumental role than the polytope size or shape. This example was a somewhat extreme demonstration that, with some tasks, it may be desirable for the robot's capabilities to be momentarily skewed one way or another. The relationship between control signal and ellipsoid (or polytope) is complex as well. Though not obvious from Figure 4.5, some actuator commands appear to drive the robot toward a desired posture, whereas others seem to reposition the polytope in anticipation of future needs. I interpret these subtle complexities, not as an inability on the part of the robot, but rather as an impediment for the design of a suitable control system. For this reason (and others) I speculate that intelligent control and machine learning techniques will become more prevalent for manipulator control.

Despite the difficulties with the design of coordinated movement, the example in Section 4.3.3 provides evidence that singular configurations may be useful in this regard. Wang *et al.* [156] also observed the benefits of singular configurations. Their work extended the payload capacity of a Puma industrial robot by formulating an optimal control problem and by solving the corresponding nonlinear optimization problem. Singular configurations were explicitly part of neither the problem nor the solution technique, yet the best trajectories routinely passed through singularities. Wang *et al.* offered the following explanation [156]: Singular configurations are beneficial because the robot structure sup-

ports heavy loads, thereby freeing the actuators to contribute increased torque for some other aspect of the task.

This interpretation is consistent with "kinematic" manipulability based on the Jacobian alone, i.e., the enhanced static force transmission near singularities. The results in this chapter, however, suggest that dynamic effects are important as well. In particular, I demonstrated that large displacements of the acceleration polytope occur for motion near singular configurations. These results, like other work on *dynamic* manipulability, describe the capabilities of the robot *and* its actuators, whereas the static force ellipsoid describes the mapping between end-effector force and net joint torque—with no explicit role for the actuators.

With regard to robot skill acquisition, this chapter provides theoretical underpinnings for the selection of via points to test in the learning framework of Chapter 3. In particular, for high-speed movements one can use the empirical approach described in Section 4.4 to find those candidate via points most compatible with the task. And for the weightlifter, this work also provides the basis for explaining the observed advantage of the reversal solution over the standard solution.

CHAPTER 5

SUPERVISED ACTOR-CRITIC REINFORCEMENT LEARNING

For robot manipulators, the standard solution to a movement problem involves, not machine learning, but rather the construction of a kinematic trajectory that becomes the input to a tracking controller. This approach often comes with associated performance guarantees, although at the price of inefficient movements that fail to exploit dynamics. The goal of this chapter is to get the best of both worlds by combining standard techniques for manipulator control with a system that learns to exploit dynamics. In particular, I describe a *supervised actor-critic architecture*, whereby a feedback controller acts as a supervisor that provides another source of information, in addition to the critic, with which to improve the actor's policy.

5.1 Combining Supervised Learning with Reinforcement Learning

Reinforcement learning (RL) and supervised learning are often portrayed as distinct methods of learning from experience. RL methods are often applied to problems involving sequential dynamics and optimization of a scalar performance objective, with online exploration of the effects of actions. Supervised learning methods, on the other hand, are frequently used for problems involving static input-output mappings and minimization of a vector error signal, with no explicit dependence on how training examples are gathered. As discussed by Jordan and Rumelhart [75] RL and supervised learning differ not so much in the *algorithms* employed but rather in the nature of the *problems* solved. The distinguishing feature is whether feedback from the environment serves as an evaluation signal or as

an error signal, and the focus of this chapter is problems where both kinds of feedback are available to a learning system *at the same time*.

As an example, consider a young child learning to throw a ball. For this motor task, as well as many others, there is no substitute for ongoing practice. The child repeatedly throws a ball under varying conditions and with variation in the executed motor commands. Bernstein [21] called this kind of trial-and-error learning "repetition without repetition." The outcome of each movement, such as the visual confirmation of whether the ball reached a nearby parent, acts as an evaluation signal that provides the child with feedback about the quality of performance—but with no specific information about what corrections should be made. In addition, the parent may interject error information in the form of verbal instruction or explicit demonstration of what went wrong with each preceding throw. In reality, the feedback may be much more subtle than this. For instance, the final position of the ball reveals some directional information, such as too far to the left or the right; a learned forward model [75] can then be used to make this corrective information more specific to the child's sensorimotor apparatus. Similarly, the tone of the parent's voice may provide evaluative praise simultaneously with the verbal error information. In any case, the two kinds of feedback play interrelated, though complementary roles [15]: The evaluation signal drives skill optimization, whereas the error signal provides a standard of correctness that helps ensure a certain level of performance, either on a trial-by-trial basis or for the learning process as a whole.

To overcome some of the difficulties with RL alone, a number of researchers have proposed the use of supervisory information that effectively transforms a learning problem into one which is easier to solve. Common examples involve shaping [53, 97, 108], learning from demonstration [77, 92, 121, 133], or the use of carefully designed controllers [72, 84, 112]. Approaches that explicitly model the role of a supervisor include ASK FOR HELP [43], RATLE [96], and the mentor framework [115]. In each case, the goal of learning is an optimal policy, i.e., a mapping from states to actions that optimizes

some performance criterion. Despite the many successful implementations, none of these approaches combines both kinds of feedback as described shortly. Either supervised learning precedes RL during a separate training phase, or else the supervisory information is used to modify a *value function* rather than a policy directly. Those methods based on Q-learning [157], for instance, build a value function that ranks the actions available in a given state. The corresponding policy is then represented implicitly, usually as the action with the best ranking for each state.

The approach taken in this chapter involves an actor-critic architecture for RL [18]. Actor-critic architectures differ from other value-based methods in that separate data structures are used for the control policy (the "actor") and the value function (the "critic"). One advantage of the actor-critic framework is that action selection requires minimal computation [138]. Actions are computed directly from the actor, whereas methods that lack a separate data structure for the policy typically require a repeated search for the action with the best value, and this search can become computationally prohibitive, especially for real-valued actions as in the examples of Section 5.3.

Another important advantage of the actor-critic framework is that the policy can be modified directly by standard supervised learning methods. In other words, the actor can change its behavior based on state-action training pairs provided by a supervisor, without the need to calculate the values of those training data. The critic (or some other comparable mechanism) is still required for optimization, whereas the supervisor helps the actor achieve a level of proficiency whenever the critic has a poor estimate of the value function. In the next section I describe a *supervised* actor-critic architecture where the supervisor supplies not only error information for the actor, but also actions for the environment.

5.2 Supervised Actor-Critic Architecture

Figure 5.1 shows a schematic of the usual actor-critic architecture [138] augmented by three major pathways for incorporating supervisor information. Along the "shaping"

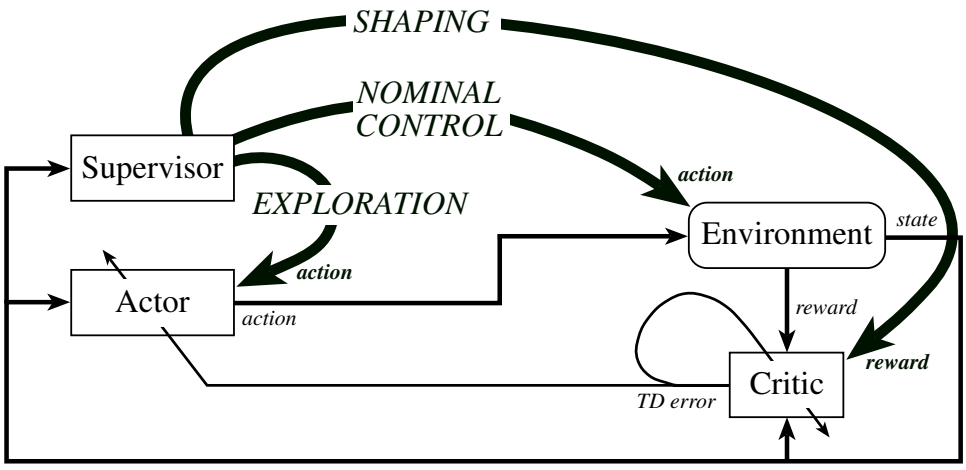

Figure 5.1. Actor-critic architecture and several pathways for incorporating supervisor information.

pathway, the supervisor supplies an additional source of evaluative feedback, or *reward*, that essentially simplifies the task faced by the learning system. For instance, the critic may receive favorable evaluations for behavior which is only approximately correct given the original task. As the actor gains proficiency, the supervisor then gradually withdraws the additional feedback to shape the learned policy toward optimality for the true task. With "nominal control" the supervisor sends control signals (*actions*) directly to the controlled system (the *environment*). For example, the supervisor may override bad commands from the actor as a way to ensure safety and to guarantee a minimum standard of performance. And along the "exploration" pathway, the supervisor provides the actor with hints about which actions may or may not be promising for the current situation, thereby altering the exploratory nature of the actor's trial-and-error learning. The focus of this section is on the latter two pathways and, in particular, on the use of supervised learning which offers a powerful counterpart to RL methods.

The combination of supervised learning with actor-critic RL was first suggested by Clouse and Utgoff [44] and independently by Benbrahim and Franklin [20]. Their approach has received almost no attention, yet the more general problem of how to combine a teacher with RL methods has become quite popular. In their work, Benbrahim and

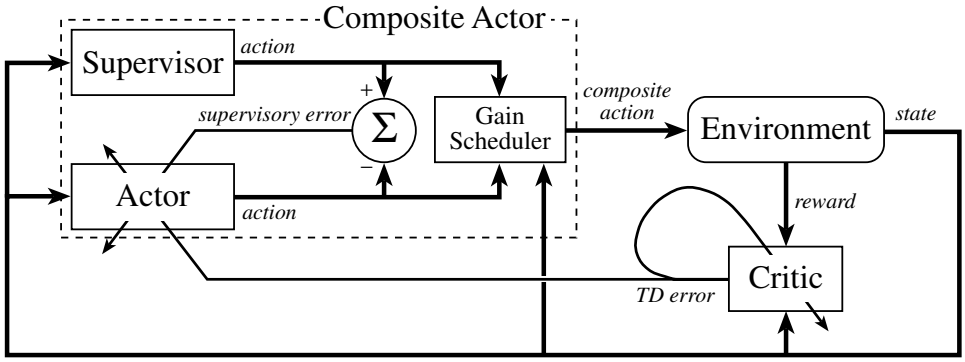

Figure 5.2. The supervised actor-critic architecture.

Franklin [20] used pre-trained controllers, called "guardians," to provide safety and performance constraints for a biped robot. Joint position commands were sent to the robot by a central pattern generator, but those commands were modified by the guardians whenever a constraint was violated. Superimposed with the joint position commands were exploratory actions generated according to Gullapalli's SRV algorithm [67]. In effect, the central pattern generator learned not only from exploration, but from the guardians as well.

Figure 5.2 shows my version of the supervised actor-critic architecture, which differs from previous work in a key way described in Section 5.2.2. Taken together, the actor, the supervisor and the gain scheduler[1] form a "composite" actor that sends a composite action to the environment. The environment responds to this action with a transition from the current state, s, to the next state, s'. The environment also provides an evaluation called the immediate reward, r. The job of the critic is to observe states and rewards and to build a value function, $V^{\pi}(s)$, that accounts for both immediate and future rewards received under the composite policy, π. This value function is defined recursively as

$$V^{\pi}(s) = \sum_{s' \in \mathcal{S}} \Pr(s'|s,a)\{R(s') + \gamma V^{\pi}(s')\},$$

[1] "Gain scheduling" refers to the construction of a global nonlinear controller by interpolation, or scheduling, of local linear controllers [126]. In this thesis I use the term in a broader sense to mean the blending of two or more sources of control actions.

where $R(s')$ is the expected value of r, $\gamma \in [0,1]$ is a factor that discounts the value of the next state, and $\Pr(s'|s,a)$ is the probability of transitioning to state s' after executing action $a = \pi(s)$. Here the focus is on deterministic policies, although this work also generalizes to the stochastic case where π represents a distribution for choosing actions probabilistically.

For RL problems, the expected rewards and the state-transition probabilities are typically unknown, and learning must proceed from samples, i.e., from observed rewards and state transitions. Temporal-difference (TD) methods [137] are commonly used to update—on a per action basis—the state-value *estimates*, $V(s)$, by an amount proportional to the TD error, defined as

$$\delta = r + \gamma V(s') - V(s).$$

Whenever the TD error is positive, the state of affairs is better than expected and so one increases the estimated value of state s. Similarly, when $\delta < 0$, the action chosen according to π resulted in a situation worse than expected and so one decreases $V(s)$. In short, TD methods improve past estimates of the value function by using future—typically more accurate—estimates of V^{π}.

5.2.1 The Gain Scheduler

For deterministic policies and real-valued actions, the gain scheduler computes the composite action, a, as simply a weighted sum of the actions given by the component policies. In particular,

$$a \leftarrow k a^E + (1-k) a^S,$$

where a^E is the actor's exploratory action and a^S is the supervisor's action, as given by policies π^E and π^S, respectively. (The supervisor's actions are observable but its policy is unknown.) Denoted by a^A is the actor's greedy action determined by the corresponding

policy, π^A. Typically, π^E is a copy of π^A modified to include an additive random variable with zero mean. Thus, each exploratory action is simply a noisy copy of the corresponding greedy action, although there is certainly the possibility of more sophisticated exploration strategies.

The parameter $k \in [0,1]$ interpolates between π^E and π^S, and therefore k determines the level of control, or autonomy, on the part of the actor.[2] In general, the value of k varies with state, although I drop the explicit dependence on s to simplify notation. The parameter k also plays an important role in modifying the actor's policy, as described in more detail below. I assume that π^A is given by a function approximator with the parameter vector w, and after each state transition, those parameters are updated according to a rule of the form

$$w \leftarrow w + k\Delta w^{\mathrm{RL}} + (1-k)\Delta w^{\mathrm{SL}}, \tag{5.1}$$

where Δw^{RL} and Δw^{SL} are the individual updates based on RL and supervised learning, respectively. Thus, k also interpolates between two styles of learning.

The use of k—a single state-dependent parameter that trades off between two sources of action and learning—allows for a wide range of interactions between actor and supervisor. If the actor has control of the gain scheduler, for instance, then the actor can set the value of k near 0 whenever it needs help from its supervisor, cf. ASK FOR HELP [43]. Similarly, if the supervisor has control of the gain scheduler, then the supervisor can set $k = 0$ whenever it loses confidence in the autonomous behavior of the actor, cf. RATLE [96]. The gain scheduler may even be under control of a third party. For example, a linear feedback controller can play the role of supervisor, and then a human operator can adjust the value of k as a way to switch between actor and supervisor, perhaps at a longer time scale than that of the primitive actions.

[2] For the stochastic case, k gives the probability that the gain scheduler chooses the actor's exploratory action rather than the supervisor's action.

5.2.2 The Actor Update Equation

To make the reinforcement-based adjustment to the parameters of π^A the actor computes

$$\Delta w^{\mathrm{RL}} \leftarrow \alpha\delta(a^E - a^A)\nabla_w \pi^A(s), \tag{5.2}$$

where α is a step-size parameter. Eq. (5.2) is similar to the update used by the *REINFORCE* class of algorithms [162], although here I utilize the gradient of the deterministic policy π^A rather than that of the stochastic exploration policy π^E. When the TD error is positive, this update will push the greedy policy evaluated at *s* closer to a^E, i.e., closer to the exploratory action which led to a state with estimated value better than expected. Similarly, when $\delta < 0$, the update will push $\pi^A(s)$ away from a^E and in subsequent visits to state *s* the corresponding exploratory policy will select this unfavorable action with reduced probability.

To compute the supervised learning update, Δw^{SL}, the actor seeks to minimize in each observed state the supervisory error

$$E = \frac{1}{2}[\pi^S(s) - \pi^A(s)]^2.$$

Locally, this is accomplished by following a steepest descent heuristic, i.e., by making an adjustment proportional to the negative gradient of the error with respect to *w*:

$$\Delta w^{\mathrm{SL}} \leftarrow -\alpha\nabla_w E(s).$$

Expanding the previous equation with the chain rule and substituting the observed actions gives the usual kind of gradient descent learning rule:

$$\Delta w^{\mathrm{SL}} \leftarrow \alpha(a^S - a^A)\nabla_w \pi^A(s). \tag{5.3}$$

Finally, by substituting Eqs. (5.2) and (5.3) into Eq. (5.1) one obtains the desired actor update equation:

$$w \leftarrow w + \alpha[k\delta(a^E - a^A) + (1-k)(a^S - a^A)]\nabla_w \pi^A(s), \tag{5.4}$$

Eq. (5.4) summarizes a steepest descent algorithm where k trades off between two sources of gradient information:[3] one from a performance surface based on the evaluation signal and one from a quadratic error surface based on the supervisory error. Table 5.1 gives a complete algorithm.

As mentioned above, the architecture shown in Figure 5.2 is similar to one suggested previously by Benbrahim and Franklin [20] and by Clouse and Utgoff [44]. However, the approach in this chapter is novel in the following way. In the figure, I show a direct connection from the supervisor to the actor, whereas the supervisor in both [20] and [44] influences the actor indirectly through its effects on the environment as well as the TD error. Using the notation herein the corresponding update equation for these other approaches, e.g., [20, Eq. (1)], essentially becomes

$$w \leftarrow w + \alpha[k\delta(a^E - a^A) + (1-k)\delta(a^S - a^A)]\nabla_w \pi^A(s) \tag{5.5}$$

$$= w + \alpha\delta[ka^E + (1-k)a^S - a^A]\nabla_w \pi^A(s). \tag{5.6}$$

The key attribute of Eq. (5.5) is that the TD error modulates the supervisory error, $a^S - a^A$. This may be a desirable feature if one "trusts" the critic more than the supervisor, in which case one should view the supervisor as an additional source of exploration. However, Eq. (5.5) may cause the steepest descent algorithm to ascend the associated error surface, especially early in the learning process when the critic has a poor estimate of the true value

[3] In practice an additional parameter may be needed to scale the TD error. This is equivalent to using two step-size parameters, one for each source of gradient information.

Table 5.1. The supervised actor-critic learning algorithm for deterministic policies and real-valued actions.

input
  critic value function, $V(s)$, parameterized by θ
  actor policy, $\pi^A(s)$, parameterized by w
  exploration size, σ
  actor step size, α, and critic step size, β
  discount factor, $\gamma \in [0,1]$
  eligibility trace decay factor, λ
initialize θ, w arbitrarily
repeat for each trial
  $e \leftarrow 0$ (clear the eligibility traces)
  $s \leftarrow$ initial state of trial
  repeat for each step of trial
    $a^A \leftarrow$ action given by $\pi^A(s)$
    $a^E \leftarrow a^A + N(0,\sigma)$
    $a^S \leftarrow$ action given by supervisor's unknown policy, $\pi^S(s)$
    $k \leftarrow$ interpolation parameter from gain scheduler
    $a \leftarrow ka^E + (1-k)a^S$
    $e \leftarrow \gamma\lambda e + \nabla_\theta V(s)$
    take action a, **observe** reward, r, and next state, s'
    $\delta \leftarrow r + \gamma V(s') - V(s)$
    $\theta \leftarrow \theta + \beta\delta e$
    $w \leftarrow w + \alpha\left[k\delta(a^E - a^A) + (1-k)(a^S - a^A)\right]\nabla_w \pi^A(s)$
    $s \leftarrow s'$
  until s is terminal

function. Moreover, when δ is small, the actor loses the ability to learn from its supervisor, whereas in Eq. (5.4) this ability depends primarily on the interpolation parameter, k.

5.3 Examples

This section presents several examples that illustrate a gradual shift from full control of the environment by the supervisor to autonomous control by the actor. In each case, the supervisor enables the composite actor in Figure 5.2 to solve the task *on the very first trial and on every trial*, while it improves, whereas the task is virtually impossible to solve with RL alone.

For each example I used the learning algorithm in Table 5.1 with step-size parameters of $\alpha = 0.1$ for the actor and $\beta = 0.3$ for the critic. To update the critic's value function, I used the TD(λ) algorithm [137] with $\lambda = 0.7$. Both actor and critic were implemented by a tile coding scheme, i.e., CMAC [3], with a total of 25 tilings, or layers, per CMAC. One advantage of tile coding schemes is that each tile acts as a binary feature, and so the algorithm in Table 5.1 was easily modified to take advantage of replacing eligibility traces [129].

5.3.1 Ship Steering Task

For pedagogical purposes and also to facilitate future comparison with other methods, I chose a standard problem from the optimal control literature where the task is to steer a ship to a goal in minimum time [34]. The ship moves at a constant speed of $C = 0.01$ km $\cdot$ s^{-1}, and the real-valued state and action are given, respectively, by the ship's two-dimensional position and scalar heading. For this problem, the supervisor is a hand-crafted controller that always steers directly toward the origin $(0, 0)$. Under full supervision, this strategy ensures that the ship will reach the goal eventually—but not in minimum time due to a water current that complicates the task. More specifically, the equations of motion are

$$\dot{x} = C\,(\cos\phi - y),\ \dot{y} = C\sin\phi, \tag{5.7}$$

where ϕ is the ship's heading. Notice that the water current acts along the horizontal direction, x, yet depends solely on the vertical position, y. The start location is $x_0 = 3.66$, $y_0 = -1.86$, and the goal region has a of radius 0.1 km centered at the origin. A convenient feature of this test problem is that one can solve for the optimal policy analytically [34], and the dark curve in Figure 5.3 shows the corresponding optimal path. Under the optimal policy the minimum time to goal is 536.6 s while the supervisor's time to goal is 1111 s.

Eq. (5.7) was integrated numerically using Euler's method with a step size of 1 s. Control decisions by the gain scheduler were made every 25 s, at which time the ship changed

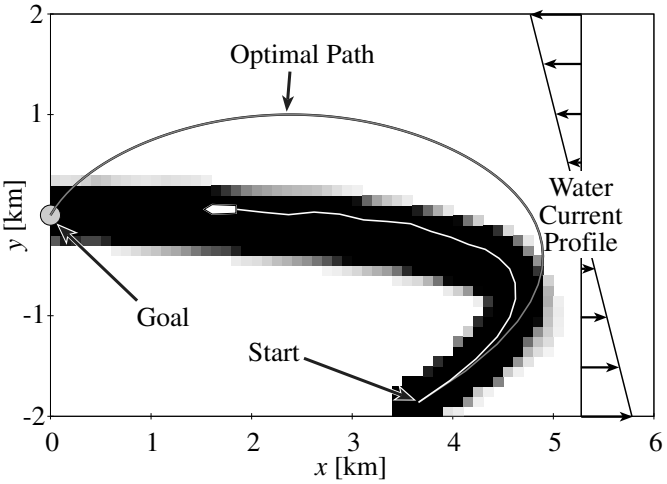

Figure 5.3. Ship steering task simulator after 1000 learning trials. The grayscale region indicates the level of autonomy, from $k = 0$ (white) to $k = 1$ (black).

heading instantaneously. Exploratory actions, a^E, were Gaussian distributed with a standard deviation of 10 degrees and a mean equal to the greedy action, a^A. The CMAC tiles were uniform with a width of 0.5 km along each input dimension. The actor CMAC was initialized to steer the ship leftward while the critic CMAC was initialized to $V(s) = -1000$, for all s. Rewards were -1 per time step, and the discount factor was $\gamma = 1$, i.e., no discounting.

To make the interaction between supervisor and actor dependent on state, the interpolation parameter, k, was set according to a state-visitation histogram, also implemented as a CMAC with 25 uniform tilings. At the end of each trial, the weights from the "visited" histogram tiles were incremented by a value of 0.0008, for a total increment of 0.02 over the 25 layers. During each step of the simulation, the value of k was set to the CMAC output for the current state, with values cut off at a maximum of $k = 1$, i.e., at full autonomy. Thus, the gain scheduler made a gradual shift from full supervision to full autonomy as the actor and critic acquired enough control knowledge to reach the goal reliably. A decay factor of 0.999 was also used to downgrade the weight of each CMAC tile; in effect, autonomy "leaked away" from infrequently visited regions of state space.

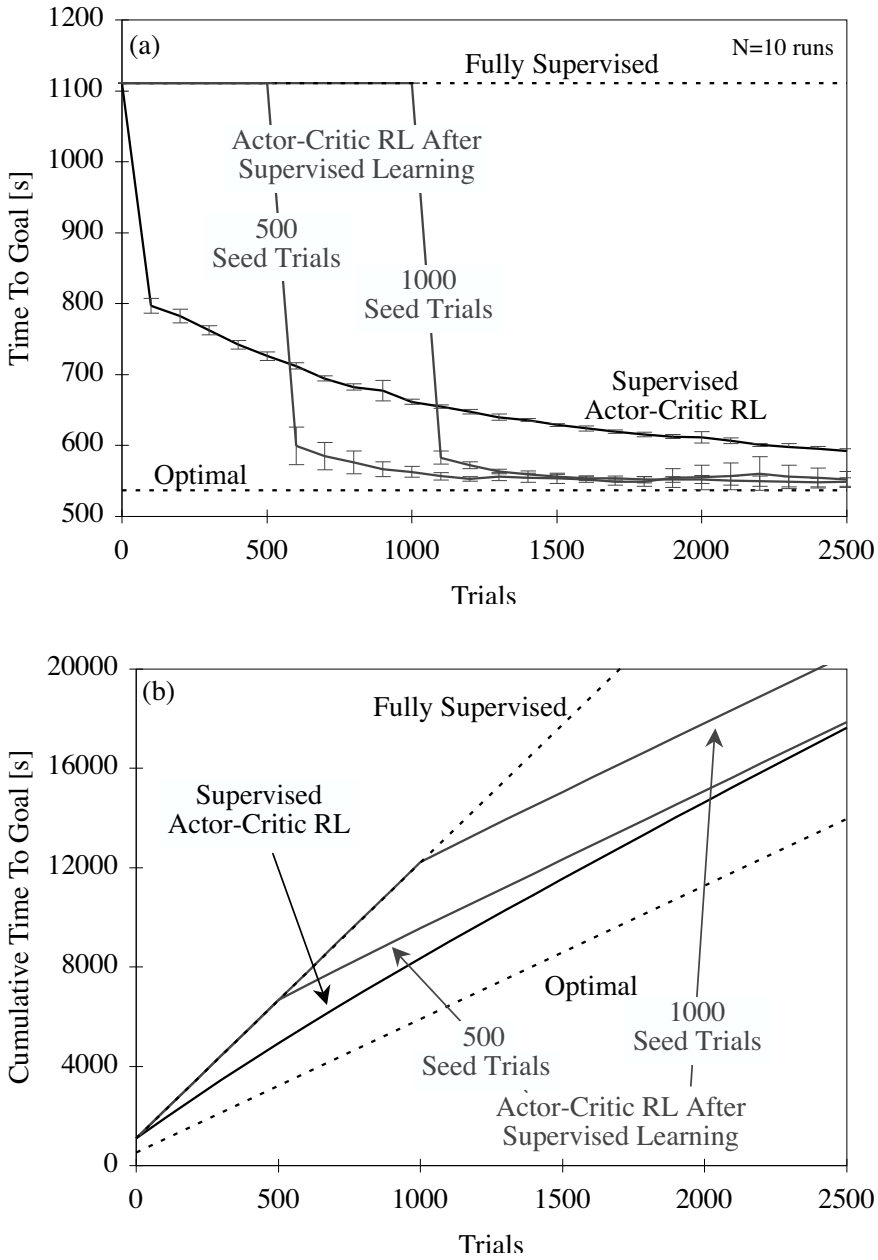

Figure 5.4. Effects of learning for the ship steering task and 10 runs of 2500 trials each: (a) time to goal, and (b) cumulative time to goal evaluated after every 100 trials of learning. For the supervised learning seed trials the initial position of the ship was chosen randomly from the region $0 \leq x \leq 6$, $-2 \leq y \leq 2$.

Figure 5.4 shows the effects of learning for each of three cases. One case corresponds to the supervised actor-critic algorithm in Table 5.1, with the parameter values described above. The other cases are from a two-phase learning process where either 500 or 1000

trials of supervised learning were used to seed the actor's policy as well as the critic's value function, followed by RL alone, cf. [77, 92, 121, 133]. That is, $k = 0$ for the first 500 or 1000 trials, and $k = 1$ thereafter. Each case shows rapid improvement during the first 100 trials of RL, followed by slower convergence toward optimality. In panel (a) the two-phase process appears to give much improved performance—if one is willing to pay the price associated with an initial learning phase that gives no immediate improvement. Indeed, if we examine cumulative reward instead, as in panel (b), the roles become reversed with the two-phase process performing worse, at least for the short-term. With 500 seed trials, the two-phase process closes the gap with continued learning and eventually outperforms the supervised actor-critic approach after 3200 trials. However, with fewer than 500 seed trials of supervised learning, the actor is able to reach the goal either unreliably (300 and 400 trials) or else not at all (100 and 200 trials).

5.3.2 Manipulator Control

My second example demonstrates that the style of control and learning used for the ship steering task is also suitable for learning to exploit the dynamics of a simulated robotic arm. The arm was modeled as a two-link pendulum with each link having length 0.5 m and mass 2.5 kg, and the equations of motion [47] were integrated numerically using Euler's method with a step size of 0.001 s. Actions from both actor and supervisor were generated every 0.75 s and were represented as two-dimensional velocity vectors with joint speed limits of ± 0.5 rad/sec. The task was to move with minimum effort from the initial configuration at

$$\mathbf{q} = \begin{bmatrix} -90 & 0 \end{bmatrix}^{\mathrm{T}} \text{ degrees}$$

to the goal configuration at

$$\mathbf{q} = \begin{bmatrix} 135 & 90 \end{bmatrix}^{\mathrm{T}} \text{ degrees.}$$

Like the weightlifting example of Chapter 3, effort was quantified as the total integrated torque magnitude.

Similar to the ship steering task, the supervisor for this example is a hand-crafted controller that moves the arm at maximum speed directly toward the goal in configuration space. Therefore, actions from the supervisor always lie on a unit square centered at the origin, whereas the actor is free to choose from the entire set of admissible actions. In effect, the supervisor's policy is to follow a straight-line path to the goal—which is time-optimal given the velocity constraints. Due to the dynamics of the robot, however, straight-line paths are not necessarily optimal with respect to other performance objectives, such as minimum energy.

A lower-level control system was responsible for transforming commanded velocities into motor torques for each joint. This occurred with a control interval of 0.001 s and in several stages: First, the commanded velocity was adjusted to account for acceleration constraints that eliminate abrupt changes in velocity, especially at the beginning and end of movement. The adjusted velocity, along with the current position, was then used to compute the desired position at the end of the next control interval. Third, a proportional-derivative controller converted this target position into joint torques, but with a target velocity of zero rather than the commanded velocity. And finally, a simplified model of the arm was used to adjust the feedback-based torque to include a feedforward term that compensates for gravity. This scheme is intended to match the way some industrial manipulators are controlled once given a higher-level movement command, e.g., velocity as used here. Gravity compensation guarantees stability of the lower-level controller [47], and the target velocity of zero helps ensure that the arm will stop safely given a communications failure with the higher level.

The above control scheme also holds an advantage for learning. Essentially, the manipulator behaves in accordance with a tracking controller—only the desired trajectory is revealed gradually with each control decision from the higher level. At this level, the ma-

nipulator behaves like an overdamped, approximately first-order system, and so policies need not account for the full state of the robot. That is, for both actor and supervisor it suffices to use reduced policies that map from positions to velocity commands, rather than policies that map from positions *and* velocities to acceleration commands. As is common with tracking controllers, this abstraction appears to do away with the dynamics exploited throughout this dissertation research. However, by designing an optimal control problem, I allow the dynamics to influence the learning system by way of the performance objective, i.e., through the reward function.

For the RL version of this optimal control problem, rewards were the negative effort accumulated over each 0.75 s decision interval, and the discount parameter was set to $\gamma = 1$. As with the ship steering task, exploratory actions, a^E, were Gaussian distributed with a mean equal to the greedy action, except that a^E was clipped at the joint speed limits. The standard deviation of the exploratory actions was initially 1.0 rad/sec, but this value decayed exponentially toward zero by a factor of 0.999 after each trial. CMAC tiles were uniform with a width of 25 degrees along each input dimension; the actor CMAC was initialized to zero whereas the critic CMAC was initialized to -300. Like the previous example, a third CMAC was used to implement a state-visitation histogram that stored the value of the interpolation parameter, k. As above, the histogram increment was 0.02 over the 25 layers and the decay factor was 0.999.

Figure 5.5(a) shows the configuration of the robot every 0.75 s along a straight-line path to the goal. The proximal joint has more distance to cover and therefore moves at maximum speed, while the distal joint moves at a proportionately slower speed. The total effort for this fully supervised policy is 258 Nm·s. Figures 5.5(b) and 5.5(c) show examples of improved performance after 5000 trials of learning, with a final cost of 229 and 228 Nm·s, respectively. In each of the left-hand diagrams, the corresponding "spokes" from the proximal link fall in roughly the same position, and so the observed improvements are due to

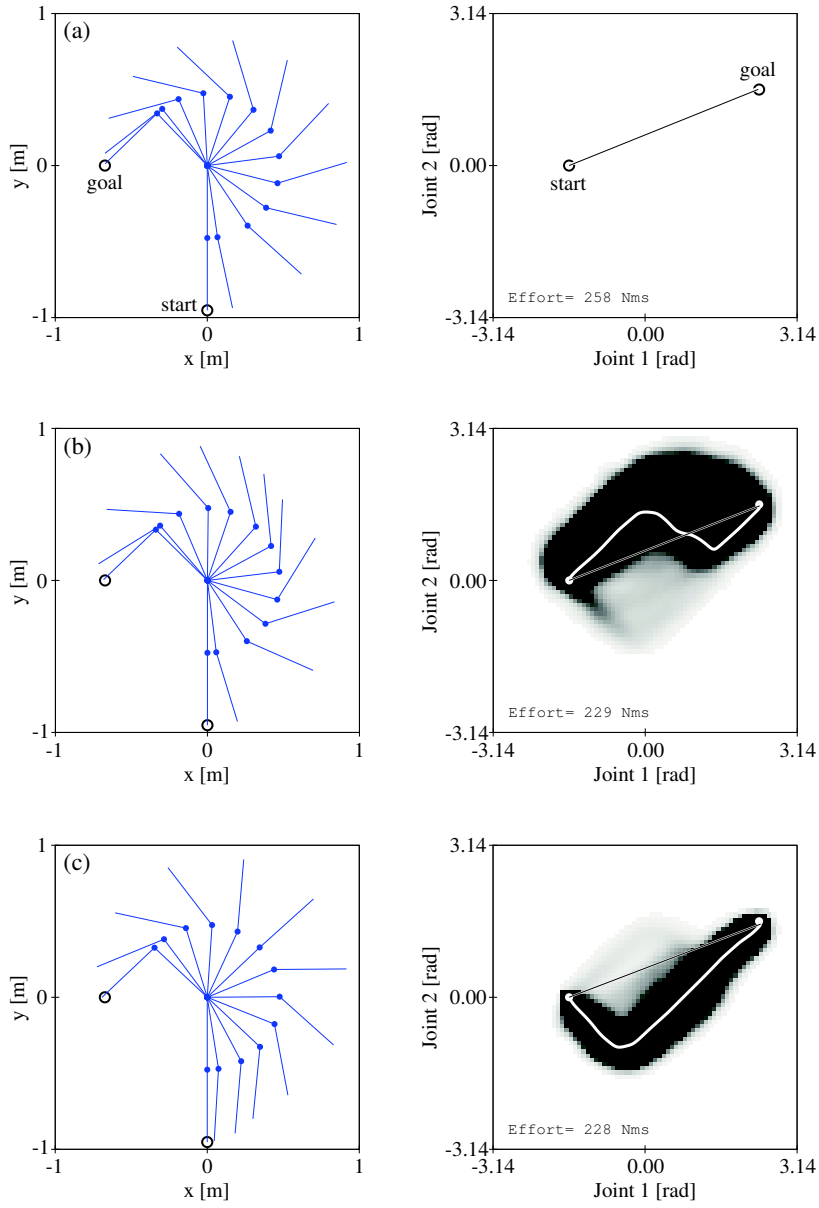

Figure 5.5. Two-link arm simulator after (a) no learning and (b,c) 5000 learning trials. Configuration-space paths after learning are shown in white, and the grayscale region indicates the level of autonomy, from $k = 0$ (white) to $k = 1$ (black).

the way the distal joint modulates its movement around the straight-line path, as shown in the right-hand diagrams.

Figure 5.6 shows the effects of learning averaged over 25 runs. The value of the optimal policy for this task is unknown, although the best observed solution has a cost of 216 Nm·s.

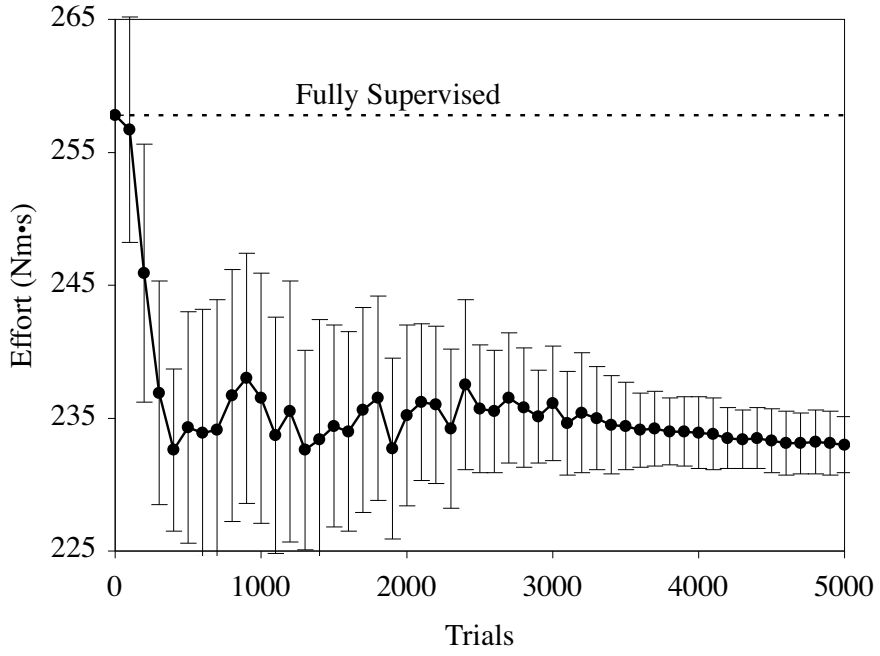

Figure 5.6. Effects of learning for the two-link arm over 25 runs of 5000 trials each.

Most improvement happens within 400 trials and the remainder of learning shows a drop in variability as the exploration policy "decays" toward the greedy policy. One difficulty with this example is the existence of many locally optimal solutions to the task. This causes the learning system to wander among solutions, with convergence to one of them only when forced to do so by the reduced exploration.

5.3.3 Case Study With a Real Robot

To demonstrate that the methods in this chapter are suitable for real robots, I replicated the previous example with a seven degree-of-freedom whole arm manipulator (WAM; Barrett Technology Inc., Cambridge, MA). Figure 5.7 shows a sequence of several postures as the WAM moves from the start configuration (far left frame) to the goal configuration (far right frame). As with the previous example the task was formulated as a minimum-effort optimal control problem—utilizing a stable tracking controller and a supervisor that generates straight-line trajectories to the goal in configuration space. The joint speed limits for this example were increased to ± 0.75 rad/sec rather than ± 0.5 rad/sec as used above.

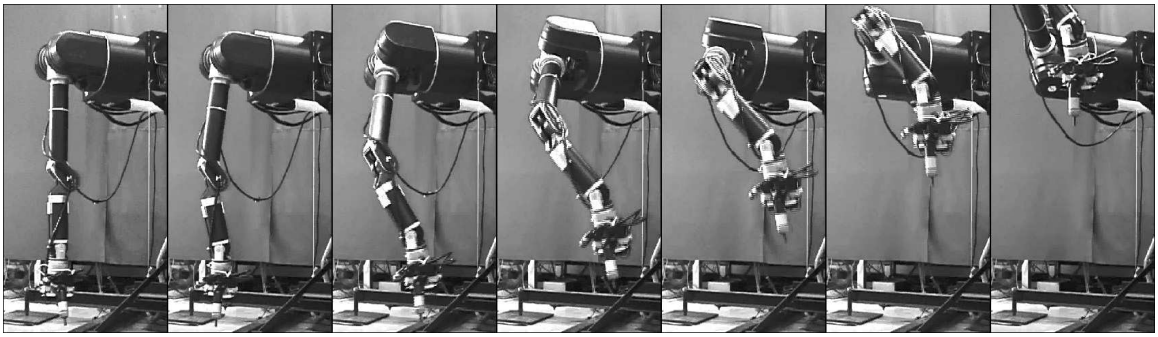

Figure 5.7. Representative configurations of the WAM after learning.

The learning algorithm was virtually identical to the one in the previous example, although several parameter values were modified to encourage reasonable improvement with very few learning trials. For instance, the histogram increment was increased from 0.02 to 0.10, thereby facilitating a faster transition to autonomous behavior. Also, the level of exploration did not decay, but rather remained constant, and a^E was Gaussian distributed with a standard deviation of 0.25 rad/sec.

Figure 5.8 shows the effects of learning averaged over 5 runs. Performance worsens during the first 10 to 20 trials due to the initialization of the actor's policy. More specifically, at the start of learning the actor's policy maps all inputs to the zero velocity vector, and so the actor cannot move the robot until it has learned how to do so from its supervisor. The drawback of this initialization scheme—along with a fast transition to autonomous behavior—is that early in the learning process the supervisor's commands become diminished when blended with the actor's near-zero commands. The effect is slower movement of the manipulator and prolonged effort while raising the arm against gravity. However, after 60 trials of learning the supervised actor-critic architecture shows statistically significant improvement ($p < 0.01$) over the supervisor alone. After 120 trials, the overall effect of learning is approximately 20% reduced effort despite an increased average movement time from 4.16 s to 4.34 s (statistically significant with $p < 0.05$).

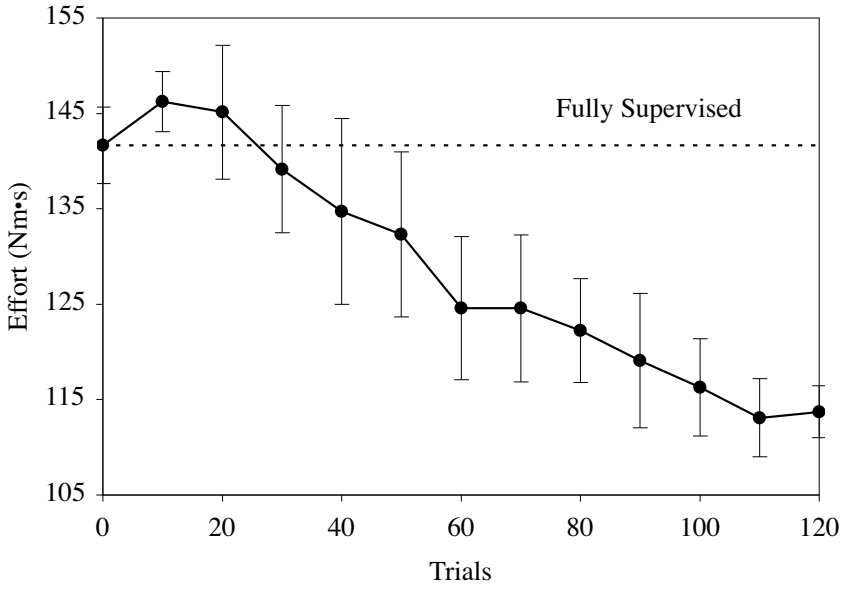

Figure 5.8. Effects of learning for the WAM over 5 runs of 120 trials each.

5.4 Discussion

As mentioned at the beginning of this chapter, one goal of this research is to exploit dynamics without throwing away the kind of information used by conventional approaches to manipulator control. The supervised actor-critic architecture provides a way to meet this goal. In particular, one can design a stable tracking controller that operates at a low level in addition to a heuristic controller at a higher level that moves the robot directly to a desired configuration. Meanwhile, the actor can improve performance by learning a policy, i.e., a family of implicit trajectories, that exploits dynamics.

The examples in Section 5.3 demonstrate a gradual shift from full supervision to full autonomy—blending two sources of actions and learning feedback. Much like the examples by Clouse [43] and by Maclin and Shavlik [96], this shift happens in a state-dependent way with the actor seeking help from the supervisor in unfamiliar territory. Unlike these other approaches, the actor also clones the supervisor's policy very quickly over the visited states. This style of learning is similar to methods that seed an RL system with training data, e.g., [121, 133], although with the supervised actor-critic architecture, the interpolation parameter allows the seeding to happen in an incremental fashion at the same time as

trial-and-error learning. Informally, the effect is that the actor knows what the supervisor knows, but only on a need-to-know basis.

One drawback of these methods for control of real robots is the time needed for training. By most standards in the RL literature, the supervised actor-critic architecture requires relatively few trials, at least for the examples presented above. However, some robot control problems may permit extremely few learning trials, say 10 or 20. Clearly, in such cases we should not expect optimality; instead we should strive for methods that provide gains commensurate with the training time. In any case, we might tolerate slow optimization if we can deploy a learning robot with provable guarantees on the worst-case performance. Recent work by Kretchmar *et al.* [84] and by Perkins and Barto [112] demonstrates initial progress in this regard.

Despite the challenges when we combine supervised learning with an actor-critic architecture, we still reap benefits from both paradigms. From actor-critic architectures we gain the ability to discover behavior that optimizes performance. From supervised learning we gain a flexible way to incorporate domain knowledge. In particular, the internal representations used by the actor can be very different from those used by the supervisor. The actor, for example, can be an artificial neural network, while the supervisor can be a conventional feedback controller, expert knowledge encoded as logical propositions, or a human supplying actions that depend on an entirely different perception of the environment's state. Moreover, the supervisor can convey intentions and solution strategies to the actor, and so this work is similar in spirit to work on imitation learning, e.g., [98, 122]. And presumably the supervisor has a certain proficiency at a given task, which the actor exploits for improved performance throughout learning.

CHAPTER 6

CONCLUSIONS

The problem of the rider has begun to
overshadow the problem of the horse.
— N.A. Bernstein, 1961

Bernstein argued in favor of motor coordination as a separate discipline, no less important than those disciplines geared toward the underlying elements of coordination. While the past 50 years have seen considerable change in the way scientists investigate biological motor coordination, only recently have roboticists begun to view coordination and skill acquisition as a problem on par with mechanism design, path planning, and lower-level feedback control. As shown throughout this dissertation, one approach to robot coordination is a combination of techniques that take advantage of strategies from biology and machine learning, without necessarily giving up the benefits of methods from mechanics and control theory.

6.1 Summary

As described in Chapter 2, control of robot manipulators typically involves trajectory planning followed by trajectory tracking. Implicit in this approach is the use of a control scheme that cancels the robot's intrinsic dynamics and that effectively decouples each joint from the others. Efficient movements that exploit dynamics are still possible within this framework, although accurate models are needed as well as more sophisticated planners that deal with dynamics and optimization.

6.1.1 Main Results

The alternative approach taken in Chapter 3 was to use a few key principles about biological motor control as strategies for designing a class of structured policies. With the right structure, a simple policy-search algorithm was then able to solve an otherwise complicated robot motor task. Moreover, this approach was effective with simulated and real robots alike.

One key ingredient of these structured policies was a gain matrix that enables individual joints to become coupled through the control system. Whereas most control engineering methods seek to diagonalize such gain matrices—so that separate controllers can be designed for each degree of freedom independently of the others—the SRS algorithm of Chapter 3 instead made use of the coupling afforded by the gain matrix. The results of learning were motor synergies that exploit dynamics while coordinating individual degrees of freedom to act as a single functional unit.

The emphasis of Chapter 4 was a theoretical analysis of how velocity-dependent dynamics affect a manipulator's performance. In particular, the velocity bias derived in Section 4.2.1 was essential for explaining the benefits of singular configurations as well as the advantages of the reversal solution to the weightlifting task. The analytical results also pointed toward an empirical approach for estimating the velocity bias but, surprisingly, without prior knowledge of velocity. In turn, this provides a means for seeding a path planner or learning algorithm with information needed to choose among qualitatively different solutions to a movement problem.

In Chapter 5, a more conventional control engineering perspective was used to design heuristic controllers guaranteed to bring a dynamic system to a goal. The supervised actor-critic architecture was then used to blend these controllers with reinforcement learning (RL) on a per action basis. Essentially, the actor relied first on supervised learning to rapidly gain a nominal level of performance, and then on RL to optimize that performance, albeit more

gradually. The framework allows one to incorporate domain knowledge very easily and to ensure success on every trial for tasks that are virtually impossible to solve with RL alone.

6.1.2 Putting It All Together

One common thread running through this dissertation is the notion of exploiting dynamics, whereby a control system takes advantage of reactive forces due to gravity, inertia and so on. The ability of the neuromuscular system to exploit dynamics is part of what separates coordination in higher animals from mere control of movement. The broad goal of this research is to imbue robots with this same ability.

In Chapter 3, this was accomplished with a biologically motivated control system along with trial-and-error learning by the SRS algorithm. Skill acquisition was driven by the optimization principle, and for the weightlifting task minimum effort was secondary to the true objective of raising a heavy payload. The best solutions passed through a singular configuration, and the results of Chapter 4 explain why singularities can be something to exploit, rather than something to avoid as is common with path planning approaches. And in Chapter 5, trial-and-error learning was used again to exploit manipulator dynamics, although this time with minimum effort as the primary objective. Skill acquisition was accomplished through supervised learning, whereas "mastery" of redundant degrees of freedom was accomplished through RL.

Another theme of this dissertation is that coordination is the outcome of a progression of learning problems. Initial solutions are relatively unrefined and inefficient, yet they set the stage for ongoing practice and optimization. As discussed in Chapter 2, for instance, excess degrees of freedom can be frozen during the early stages of skill acquisition and later released in a proximal-to-distal fashion as performance improves on the simplified task. Chapters 3 and 5 both involved a similar progression, although with alternative mechanisms for dealing with excess degrees of freedom at the start of learning.

Perhaps more to the core of this dissertation is a third common thread regarding the use of prior knowledge for a class of machine learning problems. The benefits of prior knowledge are widely appreciated, and so the more germane questions are what kinds of knowledge are useful for robot motor tasks and how do we make that knowledge part of a machine learning system? These questions were answered in Chapter 3 with crude kinematic information supplied by a coach and with subsequent encoding of that information as via points for a hierarchical control architecture. The theoretical work of Chapter 4 showed that the velocity bias offers a useful metric for evaluating the suitability of specific via points, even without prior knowledge of velocity. And in Chapter 5, controllers that were easily designed, yet sub-optimal, supplied the prior knowledge and supervised learning made that knowledge available to an RL system as needed in an online, incremental fashion.

6.2 Future Work

In Chapter 3 we saw that just one or two via points was enough to enable learning of a complicated motor skill. Then the analysis in Chapter 4 explained what we already observed empirically, i.e., the reversal solution to the weightlifting task is better than the standard solution. One bit of work for the future is to use the velocity bias to predict ahead of time which one of several candidate via points is best for a novel task. Other extensions of Chapter 4 are at least twofold: First, derive the velocity bias for both rotational and translational acceleration of the end-effector, rather than just translational acceleration as in Section 4.2. Second, replace the empirical approach of Section 4.4 with an analytical characterization of how the velocity bias changes with posture. The desideratum is an analogue of the manipulability ellipsoid—one that describes the way arbitrary joint-space velocities are mapped to displacements of the ellipsoid.

Another line of future work involves the framework presented in Chapter 3. One hypothesis left untested is that a small number of via points, followed by trial-and-error learn-

ing, is an approach to robot coordination which is suitable for a much larger class of motor problems than simply weightlifting. More generally, a standard motion planning algorithm could supply a small number of via points along a collision-free path, and then learning could be used to transform this kinematic solution into one that exploits dynamics. Other opportunities include imitation learning as the means for identifying via points.

A different, perhaps more interesting extension of this framework is in the direction of biology. For example, one possibility is to replace the linear proportional-derivative controllers of Section 3.4 with a biologically plausible model of equilibrium-point control, such as Feldman's λ model [58]. Additional modifications, e.g., feedback delay, may eventually lead to a neurobiological model of skill acquisition. Another possibility is to abstract away from neurons and use the framework of Chapter 3 as a behavioral model of human skill acquisition, at least for certain movements. Handwriting is a candidate skill that could be explained by a relatively small number of via points that capture the surface form of writing, and by practice, i.e., learning, as the means to graceful and efficient strokes.

For the supervised actor-critic architecture of Chapter 5, one worthwhile extension is to specify those conditions which lead to provably stable learning. As a starting point, a provably stable controller—like the ones used in Section 5.3—can act as supervisor throughout learning. But for stability of the supervisor and actor together, constraints must be placed on exploration and on the transfer from full supervision to full autonomy.

From a machine learning perspective, perhaps the most promising line of future work is a hybrid learning algorithm that combines direct policy search with supervised learning and value-based RL. One possibility that could speed up learning is to use a value function to modify the distribution of samples drawn by the SRS algorithm. Supervised learning with a hand-crafted controller could improve the SRS algorithm in this same way. Indeed, a supervisor that deals with safety constraints could be of great benefit for the robot weightlifter, which experienced numerous failures throughout learning, especially with the heaviest payloads.

Another possibility for hybrid learning is to use direct policy search as a bias for value-based RL. For instance, policy search could provide a means for structuring the choice of exploratory actions—beyond the usual random choice and in a way more compatible with a given learning problem. By combining various styles of machine learning, one could achieve faster and more reliable performance than possible with any single method, thereby making intelligent control a practical alternative to more conventional methods.

6.3 Concluding Remarks

Robots are a remarkable example of human ingenuity. They work tirelessly on assembly lines, they enter hazardous environments without hesitation, and they perform feats of strength and control beyond anything dreamed up by Mother Nature. But current applications only scratch the surface of the potential benefits that robots hold for society. Although our capacity to design dexterous machines as grown considerably since the 1960s when the first industrial robots appeared, our ability to control them in a coordinated fashion is another matter.

In this dissertation I approached the problem of robot motor coordination from several different perspectives, and one particular emphasis was the connection with biological motor control. I demonstrated that biology has something practical to offer robotics, and I also touched briefly on the reverse, i.e., how robots can help us study biology. But perhaps a more forward looking view of this research is that it provides a foundation for coordinated human-robot interaction. Exploiting dynamics can help in this endeavor as well, but not just by making the robot faster, smoother, and more energy efficient. Dynamics are also something to take advantage of when designing the user interface for teleoperators and for assistive technology such as prosthetic limbs. The ultimate goal is a magical robotic device by today's standards—one that responds to human gestures and neuromuscular commands as if able to read its operator's mind, perhaps literally someday.

BIBLIOGRAPHY

[1] C. Abdallah, D. Dawson, P. Dorato, and M. Jamshidi. Survey of robust control for rigid robots. *IEEE Control Systems Magazine,* 11(2):24–30, 1991.

[2] J. A. Adams. A closed-loop theory of motor learning. *Journal of Motor Behavior*, 3 (2):111–150, 1971.

[3] J. S. Albus. *Brains, Behavior, and Robotics*. Byte Books, Peterborough, NH, 1981.

[4] C. H. An, C. G. Atkeson, and J. M. Hollerbach. *Model-Based Control of a Robot Manipulator*. The MIT Press, Cambridge, MA, 1988.

[5] C. W. Anderson. Approximating a policy can be easier than approximating a value function. Technical Report CS-00-101, Colorado State University, 2000.

[6] R. W. Anderson. Biased random-walk learning: a neurobiological correlate to trial-and-error. In O. Omidvar and J. Dayhoff, editors, *Neural Networks and Pattern Recognition*, pages 221–244. Academic Press, San Diego, CA, 1998.

[7] S. Arimoto, S. Kawamura, and F. Miyazaki. Bettering operation of robots by learning. *Journal of Robotic Systems*, 1(2):123–140, 1984.

[8] R. C. Arkin. *Behavior-Based Robotics*. The MIT Press, Cambridge, MA, 1998.

[9] H. Asada. A geometrical representation of manipulator dynamics and its application to arm design. *Journal of Dynamic Systems, Measurement, and Control*, 105:131–135,142, 1983.

[10] H. Asada and H. Izumi. Automatic program generation from teaching data for the hybrid control of robots. *IEEE Transactions on Robotics and Automation*, 5(2): 166–173, 1989.

[11] C. G. Atkeson and J. McIntyre. Robot trajectory learning through practice. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1737–1742, 1986. IEEE, Piscataway, NJ.

[12] C. G. Atkeson and S. Schaal. Robot learning from demonstration. In D. H. Fisher, editor, *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 12–20, 1997. Morgan Kaufmann, San Francisco, CA.

[13] L. Baird and A. Moore. Gradient descent for general reinforcement learning. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances In Neural Information Processing Systems 11*, pages 968–974, 1999. The MIT Press, Cambridge, MA.

[14] A. G. Barto. Connectionist learning for control. In W. T. Miller, R. S. Sutton, and P. J. Werbos, editors, *Neural Networks for Control*, pages 5–58. The MIT Press, Cambridge, MA, 1990.

[15] A. G. Barto. Reinforcement learning in motor ontrol. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 809–813. The MIT Press, Cambridge, MA, 1995.

[16] A. G. Barto, A. H. Fagg, N. Sitkoff, and J. C. Houk. A cerebellar model of timing and prediction in the control of reaching. *Neural Computation*, 11(3):565–594, 1999.

[17] A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems: Theory and Applications*, in press.

[18] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:835–846, 1983.

[19] J. Baxter and P. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

[20] H. Benbrahim and J. A. Franklin. Biped dynamic walking using reinforcement learning. *Robotics and Autonomous Systems*, 22:283–302, 1997.

[21] N. A. Bernstein. *The Co-ordination and Regulation of Movements*. Pergamon Press, Oxford, 1967.

[22] N. A. Bernstein. On dexterity and its development. In M. L. Latash and M. T. Turvey, editors, *Dexterity and Its Development*, pages 1–244. Lawrence Erlbaum Associates, Mahwah, NJ, 1996.

[23] N. Berthier, R. Clifton, D. McCall, and D. Robin. Proximodistal structure of early reaching in human infants. *Experimental Brain Research*, 127:259–269, 1999.

[24] G. S. G. Beveridge and R. S. Schechter. *Optimization: Theory and Practice*. McGraw-Hill Book Co., New York, 1970.

[25] E. Bizzi, N. Hogan, F. A. Mussa-Ivaldi, and S. Giszter. Does the nervous system use equilibrium-point control to guide single and multiple joint movements? *Behavioral and Brain Sciences*, 15:501–613, 1992.

[26] R. Bongaardt and O. G. Meijer. Bernstein's theory of movement behavior: Historical development and contemporary relevance. *Journal of Motor Behavior*, 32(1):57–71, 2000.

[27] C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1-2):49–107, 2000.

[28] A. Bowling and O. Khatib. The motion isotropy hypersurface: a characterization of acceleration capability. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 965–971, 1998.

[29] M. Brady. Artificial intelligence and robotics. *Artificial Intelligence*, 26:79–121, 1985.

[30] B. Bril and Y. Breniere. Postural requirements and progression velocity in young walkers. *Journal of Motor Behavior*, 24(1):105–116, 1992.

[31] T. A. Brisson and C. Alain. A comparison of two references for using knowledge of performance in learning a motor task. *Journal of Motor Behavior*, 29(4):339–350, 1997.

[32] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2:14–23, 1986.

[33] V. B. Brooks. *The Neural Basis of Motor Control*. Oxford University Press, New York, 1986.

[34] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control*. Hemisphere Publishing Corp., New York, 1975. Revised printing by Taylor & Francis Publishers (Bristol, PA) in 1981.

[35] E. Burdet, A. Codourey, and L. Rey. Experimental evaluation of nonlinear adaptive controllers. *IEEE Control Systems Magazine*, 18(2):39–47, 1998.

[36] R. W. Byrne and A. E. Russon. Learning by imitation: a hierarchical approach. *Behavioral and Brain Sciences*, 21(5):667–683, 1998.

[37] X.-R. Cao. From perturbation analysis to Markov decision processes and reinforcement learning. *Discrete Event Dynamic Systems: Theory and Applications*, to appear.

[38] P. Chiacchio. Exploiting redundancy in minimum-time path following robot control. In *Proceedings of the 1990 American Control Conference*, volume 3, pages 2313–2318, 1990.

[39] P. Chiacchio. A new dynamic manipulability ellipsoid for redundant manipulators. *Robotica*, 18(4):381–387, 2000.

[40] P. Chiacchio, S. Chiaverini, L. Sciavicco, and B. Siciliano. Influence of gravity on the manipulability ellipsoid for robot arms. *Journal of Dynamic Systems, Measurement, and Control*, 114(4):723–727, 1992.

[41] S. L. Chiu. Task compatibility of manipulator postures. *The International Journal of Robotics Research*, 7(5):13–21, 1988.

[42] M. K. Ciliz and K. S. Narendra. Adaptive control of robotic manipulators using multiple models and switching. *The International Journal of Robotics Research*, 15 (6):592–610, 1996.

[43] J. A. Clouse. *On Integrating Apprentice Learning and Reinforcement Learning*. PhD thesis, University of Massachusetts, Amherst, 1996.

[44] J. A. Clouse and P. E. Utgoff. A teaching method for reinforcement learning. In *Proceedings of the Nineth International Conference on Machine Learning*, pages 92–101, 1992. Morgan Kaufmann, San Francisco, CA.

[45] J. J. Craig. Adaptive control of manipulators through repeated trials. In *Proceedings of the American Control Conference*, pages 1566–1573, 1984.

[46] J. J. Craig. *Adaptive Control of Mechanical Manipulators*. Addison-Wesley Publishing Company, Reading, MA, 1988.

[47] J. J. Craig. *Introduction To Robotics : Mechanics and Control*. Addison-Wesley Publishing Company, Reading, MA, 1989.

[48] L. S. Crawford. *Learning Control of Complex Skills*. PhD thesis, University of California at Berkeley, 1998.

[49] M. P. Deisenroth and K. K. Krishnan. On-line programming. In S. Y. Nof, editor, *Handbook of Industrial Robotics, Second Edition*, pages 337–351. John Wiley & Sons, Inc., New York, 1999.

[50] J. Denavit and R. S. Hartenberg. A kinematic notation for lower pair mechanisms based on matrices. *Journal of Applied Mechanics*, 22:215–221, 1955.

[51] M. Desmurget and S. Grafton. Forward modeling allows feedback control for fast reaching movements. *Trends in Cognitive Sciences*, 4(11):423–431, 2000.

[52] T. G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[53] M. Dorigo and M. Colombetti. Robot shaping: developing autonomous agents through learning. *Artificial Intelligence*, 71(2):321–370, 1994.

[54] K. L. Doty, C. Melchiorri, E. M. Schwartz, and C. Bonivento. Robot manipulability. *IEEE Transactions on Robotics and Automation*, 11(3):462–468, 1995.

[55] T. A. Easton. On the normal use of reflexes. *American Scientist*, 60:591–599, 1972.

[56] S. E. Engelbrecht. Minimum principles in motor control. *Journal of Mathematical Psychology*, in press, 1999.

[57] I. M. Feigenberg and L. P. Latash. N. A. Bernstein: The reformer of neuroscience. In M. L. Latash and M. T. Turvey, editors, *Dexterity and Its Development*, pages 247–275. Lawrence Erlbaum Associates, Mahwah, NJ, 1996.

[58] A. G. Feldman. Once more on the equilibrium-point hypothesis (λ model) for motor control. *Journal of Motor Behavior*, 18:17–54, 1986.

[59] P. M. Fitts and M. I. Posner. *Human Performance*. Greenwood Press, Westport, CT, 1967.

[60] T. Flash and N. Hogan. Optimization principles in motor control. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 682–685. The MIT Press, Cambridge, MA, 1995.

[61] C. R. Gallistel. *The Organization of Action: A New Synthesis*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1980.

[62] E. Gat. On three-layer architectures. In D. Kortenkamp, R. P. Bonasso, and R. Murphy, editors, *Artificial Intelligence and Mobile Robots*, pages 195–210. AAAI Press, Menlo Park, CA, 1998.

[63] H. Gomi and M. Kawato. Neural network control for a closed-loop system using feedback-error-learning. *Neural Networks*, 6(7):933–946, 1993.

[64] T. J. Graettinger and B. H. Krogh. The acceleration radius: a global performance measure for robotic manipulators. *IEEE Journal of Robotics and Automation*, 4(1): 60–69, 1988.

[65] P. H. Greene. Problems of organization of motor systems. In R. Rosen and F. M. Snell, editors, *Progress In Theoretical Biology*, volume 2, pages 303–338. Academic Press, New York, 1972.

[66] P. H. Greene. Why is it easy to control your arms? *Journal of Motor Behavior*, 14 (4):260–286, 1982.

[67] V. Gullapalli. A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, 3(6):671–692, 1990.

[68] V. S. Gurfinkel and P. J. Cordo. The scientific legacy of Nikolai Bernstein. In M. L. Latash, editor, *Progress in Motor Control: Bernstein's Traditions in Movement Studies*, pages 1–19. Human Kinetics, Champaign, IL, 1998.

[69] N. Hogan. Impedance control: an approach to manipulation. *Journal of Dynamic Systems, Measurement, and Control*, 107:1–24, 1985.

[70] J. M. Hollerbach and K. C. Suh. Redundancy resolution of manipulators through torque optimization. *IEEE Journal of Robotics and Automation*, 3(4):308–316, 1987.

[71] R. Horowitz. Learning control of robot manipulators. *ASME Journal of Dynamic Systems, Measurement and Control*, 115(2):402–411, 1993.

[72] M. Huber and R. A. Grupen. A feedback control structure for on-line learning tasks. *Robotics and Autonomous Systems*, 22(3–4):303–315, 1997.

[73] S. Jaric and M. L. Latash. Learning a pointing task with a kinematically redundant limb: Emerging synergies and patterns of final position variability. *Human Movement Science*, 18:819–838, 1999.

[74] M. I. Jordan. Motor learning and the degrees of freedom problem. In M. Jeannerod, editor, *Attention and Performance*, volume 13, pages 796–836. Lawrence Erlbaum Associates, Hillsdale, NJ, 1990.

[75] M. I. Jordan and D. E. Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16(3):307–354, 1992.

[76] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.

[77] M. Kaiser and R. Dillmann. Building elementary robot skills from human demonstration. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2700–2705, 1996. IEEE, Piscataway, NJ.

[78] S. W. Keele. Movement control in skilled motor performance. *Psychological Bulletin*, 70(6):387–403, 1968.

[79] J. Kelso, D. L. Southard, and D. Goodman. On the nature of human interlimb coordination. *Science*, 203:1029–1031, 1979.

[80] H. Kimura and S. Kobayashi. An analysis of actor/critic algorithms using eligibility traces: reinforcement learning with imperfect value functions. In J. W. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 278–286, 1998. Morgan Kaufmann, San Francisco, CA.

[81] D. E. Kirk. *Optimal Control Theory: An Introduction*. Prentice-Hall Inc., Englewood Cliffs, NJ, 1970.

[82] R. M. Kohl and H. A. Ben-David. (Non)issues of infinite regress in modeling motor behavior. *Journal of Motor Behavior*, 30(1):94–96, 1998.

[83] V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In S. A. Solla, T. K. Leen, and K.-R. Muller, editors, *Advances In Neural Information Processing Systems 12*, pages 1008–1014, 2000. The MIT Press, Cambridge, MA.

[84] R. M. Kretchmar, P. M. Young, C. W. Anderson, D. C. Hittle, M. L. Anderson, C. C. Delnero, and J. Tu. Robust reinforcement learning control with static and dynamic stability. *International Journal of Robust and Nonlinear Control*, 11:1469–1500, 2001.

[85] K. Kreutz. On manipulator control by exact linearization. *IEEE Transactions on Automatic Control*, 34(7):763–767, 1989.

[86] Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching: extracting reusable task knowledge from visual observation of human performance. *IEEE Transactions on Robotics and Automation*, 10(6):799–822, 1994.

[87] A. Ledebt, B. Bril, and Y. Breniere. The build-up of anticipatory behaviour—an analysis of the development of gait initiation in children. *Experimental Brain Research*, 120(1):9–17, 1998.

[88] W. Levine and G. Loeb. The neural control of limb movement. *IEEE Control Systems Magazine*, 12(6):38–47, 1992.

[89] F. L. Lewis, C. T. Abdallah, and D. M. Dawson. *Control of Robot Manipulators*. Macmillan Publishing Company, New York, 1993.

[90] Z.-M. Li, M. L. Latash, and V. M. Zatsiorsky. Force sharing among fibers as a model of the redundancy problem. *Experimental Brain Research*, 119(3):276–286, 1998.

[91] F. Lin and R. D. Brandt. An optimal control approach to robust control of robot manipulators. *IEEE Transactions on Robotics and Automation*, 14(1):69–77, 1998.

[92] L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3–4):293–321, 1992.

[93] T. Lozano-Perez. Spatial planning: a configuration space approach. *IEEE Transactions on Computers*, 32:108–120, 1983.

[94] J. Y. S. Luh. Conventional controller design for industrial robots—a tutorial. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(3):298–316, 1983.

[95] R. Luus and T. Jaakola. Optimization by direct search and systematic reduction of the size of the search region. *AIChE J.*, 19:760–766, 1973.

[96] R. Maclin and J. W. Shavlik. Creating advice-taking reinforcement learners. *Machine Learning*, 22((1-3)):251–281, 1996.

[97] M. J. Mataric. Reward functions for accelerated learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 181–189, 1994. Morgan Kaufmann, San Francisco, CA.

[98] M. J. Mataric. Sensory-motor primitives as a basis for imitation: linking perception to action and biology to robotics. In C. Nehaniv and K. Dautenhahn, editors, *Imitation in Animals and Artifacts*. The MIT Press, Cambridge, MA, 2000.

[99] P. V. McDonald, R. E. A. van Emmerik, and K. M. Newell. The effects of practice on limb kinematics in a throwing task. *Journal of Motor Behavior*, 21(3):245–264, 1989.

[100] T. A. McMahon. *Muscles, Reflexes, and Locomotion*. Princeton University Press, Princeton, New Jersey, 1984.

[101] H. Miyamoto, S. Schaal, F. Gandolfo, H. Gomi, Y. Koike, R. Osu, E. Nakano, Y. Wada, and M. Kawato. A kendama learning robot based on bi-directional theory. *Neural Networks*, 9(8):1281–1302, 1996.

[102] D. E. Moriarty, A. C. Schultz, and J. J. Grefenstette. Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11:241–276, 1999.

[103] J. Morimoto and K. Doya. Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. In P. Langley, editor, *Proceedings of The Seventeenth International Conference on Machine Learning*, 2000. Morgan Kaufmann, San Francisco.

[104] F. A. Mussa-Ivaldi and S. F. Giszter. Vector field approximation: a computational paradigm for motor control and learning. *Biological Cybernetics*, 67:491–500, 1992.

[105] S. Muthuswamy and S. Manoochehri. Optimal path planning for robot manipulators. *Journal of Mechanical Design*, 114:586–595, 1992.

[106] K. S. Narendra. Intelligent control. *IEEE Control Systems Magazine*, 11(1):39–40, 1991.

[107] K. S. Narendra, J. Balakrishnan, and M. K. Ciliz. Adaptation and learning using multiple models, switching and tuning. *IEEE Control Systems Magazine*, 15(3): 37–51, 1995.

[108] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and applications to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 278–287, 1999. Morgan Kaufmann, San Francisco, CA.

[109] R. Ortega and M. W. Spong. Adaptive motion control of rigid robots: a tutorial. *Automatica*, 25(6):877–888, 1989.

[110] M. Pandy, B. Garner, and F. Anderson. Optimal control of non-ballistic movements: a constraint-based performance criterion for rising from a chair. *Journal of Biomechanical Engineering*, 117(1):15–26, 1995.

[111] R. Parr and S. Russell. Reinforcement learning with hierarchies of machines. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 1043–1049, 1998. The MIT Press, Cambridge, MA.

[112] T. J. Perkins and A. G. Barto. Lyapunov design for safe reinforcement learning. *Journal of Machine Learning Research*, 3:803–832, 2002.

[113] R. W. Pew. Acquisition of hierarchical control over the temporal organization of a skill. *Journal of Experimental Psychology*, 71(5):764–771, 1966.

[114] E. C. Poulton. On prediction in skilled movements. *Psychological Bulletin*, 54(6): 467–473, 1957.

[115] B. Price and C. Boutilier. Implicit imitation in multiagent reinforcement learning. In I. Bratko and S. Dzeroski, editors, *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 325–334, 1999. Morgan Kaufmann, San Francisco, CA.

[116] J. Randlov, A. G. Barto, and M. T. Rosenstein. Combining reinforcement learning with a local control algorithm. In P. Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 775–782, 2000. Morgan Kaufmann, San Francisco.

[117] T. RayChaudhuri, L. G. C. Hamey, and R. D. Bell. From conventional control to autonomous intelligent methods. *IEEE Control Systems Magazine*, 16(5):78–84, 1996.

[118] D. A. Rosenbaum. *Human Motor Control*. Academic Press, New York, 1991.

[119] H. G. Sage, M. F. D. Mathelin, and E. Ostertag. Robust control of robot manipulators: a survey. *International Journal of Control*, 72(16):1498–1522, 1999.

[120] G. Sahar and J. M. Hollerbach. Planning of minimum-time trajectories for robot arms. *The International Journal of Robotics Research*, 5(3):90–100, 1986.

[121] S. Schaal. Learning from demonstration. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances In Neural Information Processing Systems 9*, pages 1040–1046, 1997. The MIT Press, Cambridge, MA.

[122] S. Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Science*, 3:233–242, 1999.

[123] S. Schaal, S. Kotosaka, and D. Sternad. Nonlinear dynamical systems as movement primitives. In *Proceedings of The First IEEE-RAS International Conference on Humanoid Robots*, 2000. Published electronically.

[124] R. A. Schmidt. A schema theory of discrete motor skill learning. *Psychological Review*, 82(4):225–260, 1975.

[125] R. A. Schmidt and T. D. Lee. *Motor Control and Learning: A Behavioral Emphasis*. Human Kinetics, Champaign, IL, 1999.

[126] J. S. Shamma. Linearization and gain-scheduling. In W. S. Levine, editor, *The Control Handbook*, pages 388–396. CRC Press, Boca Raton, FL, 1996.

[127] C. S. Sherrington. *The Integrative Action of the Nervous System*. Yale University Press, New Haven, CT, 1906.

[128] K. G. Shin and N. D. McKay. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control*, 30(6):531–541, 1985.

[129] S. P. Singh and R. S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22(1–3):123–158, 1996.

[130] M. Skubic and R. A. Volz. Acquiring robust, force-based assembly skills from human demonstration. *IEEE Transactions on Robotics and Automation*, 16(6):772–781, 2000.

[131] J.-J. E. Slotine. The robust control of robot manipulators. *The International Journal of Robotics Research*, 4(2):49–64, 1985.

[132] J.-J. E. Slotine. Putting physics in control—the example of robotics. *IEEE Control Systems Magazine*, 8(6):12–17, 1988.

[133] W. D. Smart and L. P. Kaelbling. Effective reinforcement learning for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3404–3410, 2002. IEEE, Piscataway, NJ.

[134] M. W. Spong. The swing up control problem for the acrobot. *IEEE Control Systems Magazine*, 15(1):49–55, 1995.

[135] M. W. Spong and M. Vidyasagar. *Robot Dynamics and Control*. Wiley, New York, 1989.

[136] P. Stone. *Layered Learning in Multi-Agent Systems*. PhD thesis, Carnegie Mellon University, 1998.

[137] R. S. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44, 1988.

[138] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 1998.

[139] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. A. Solla, T. K. Leen, and K.-R. Muller, editors, *Advances In Neural Information Processing Systems 12*, pages 1057–1063, 2000. The MIT Press, Cambridge, MA.

[140] R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.

[141] W. H. Swann. Direct search methods. In W. Murray, editor, *Numerical Methods For Unconstrained Optimization*, pages 13–28. Academic Press, New York, 1972.

[142] E. L. Thorndike. *Animal Intelligence*. Hafner, Darien, CT, 1911.

[143] E. Todorov and M. I. Jordan. Optimal feedback control as a theory of motor coordination. *Nature Neuroscience*, 5:1226–1235, 2002.

[144] G. Towell and J. Shavlik. Knowledge-based artificial neural networks. *Artificial Intelligence*, 70(1–2):119–165, 1994.

[145] M. T. Turvey. Preliminaries to a theory of action with reference to vision. In R. Shaw and J. Bransford, editors, *Perceiving, Acting, and Knowing*, pages 211–265. Lawrence Erlbaum Associates, Hillsdale, NJ, 1977.

[146] M. T. Turvey. Coordination. *American Psychologist*, 45(8):938–953, 1990.

[147] Y. Uno, M. Kawato, and R. Suzuki. Formation and control of optimal trajectory in human multijoint arm movement: minimum torque-change model. *Biological Cybernetics*, 61:89–101, 1989.

[148] R. E. A. van Emmerik. Kinematic adaptations to perturbations as a function of practice in rhythmic drawing movements. *Journal of Motor Behavior*, 24(1):117–131, 1992.

[149] G. J. van Ingen Schenau and A. J. van Soest. On the biomechanical basis of dexterity. In M. L. Latash and M. T. Turvey, editors, *Dexterity and Its Development*, pages 305–338. Lawrence Erlbaum Associates, Mahwah, NJ, 1996.

[150] G. J. van Ingen Schenau, A. J. van Soest, F. J. M. Gabreels, and M. W. I. M. Horstink. The control of multi-joint movements relies on detailed internal representations. *Human Movement Science*, 14(4–5):511–538, 1995.

[151] B. Vereijken, R. E. A. van Emmerik, H. T. A. Whiting, and K. M. Newell. Free(z)ing degrees of freedom in skill acquisition. *Journal of Motor Behavior*, 24(1):133–142, 1992.

[152] S. Vijayakumar and S. Schaal. Locally weighted regression: an *O(n)* algorithm for incremental real time learning in high dimensional space. In P. Langley, editor, *Proceedings of The Seventeenth International Conference on Machine Learning*, pages 1079–1086, 2000. Morgan Kaufmann, San Francisco.

[153] P. Viviani and C. Terzuolo. Trajectory determines movement dynamics. *Neuroscience*, 7(2):431–437, 1982.

[154] M. W. Walker and D. E. Orin. Efficient dynamic computer simulation of robotic mechanisms. *ASME Journal of Dynamic Systems, Measurement and Control*, 104: 205–211, 1982.

[155] C.-Y. E. Wang, W. K. Timoszyk, and J. E. Bobrow. Weightlifting motion planning for a puma 762 robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 480–485, 1999. IEEE, Piscataway, NJ.

[156] C.-Y. E. Wang, W. K. Timoszyk, and J. E. Bobrow. Payload maximization for open chained manipulators: finding weightlifting motions for a Puma 762 robot. *IEEE Transactions on Robotics and Automation*, 17(2):218–224, 2001.

[157] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3/4):279–292, 1992.

[158] P. Weiss. Self-differentiation of the basic patterns of coordination. *Comparative Psychology Monographs*, 17(4), 1941.

[159] S. D. Whitehead. A complexity analysis of cooperative mechanisms in reinforcement learning. In *Proceedings of the Nineth National Conference on Artificial Intelligence*, pages 607–613, 1991. AAAI Press, Menlo Park, CA.

[160] D. E. Whitney. The mathematics of coordinated control of prosthetic arms and manipulators. *Journal of Dynamic Systems, Measurement and Control*, pages 303–309, 1972.

[161] N. Wiener. *Cybernetics*. The MIT Press, Cambridge, MA, 1961.

[162] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[163] M. M. Williamson. Neural control of rhythmic arm movements. *Neural Networks*, 11(7-8):1379–1394, 1999.

[164] D. M. Wolpert and M. Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11:1317–1329, 1998.

[165] T. Yoshikawa. Dynamic manipulability of robot manipulators. *Journal of Robotic Systems*, 2(1):113–124, 1985.

[166] T. Yoshikawa. Manipulability of robot mechanisms. *The International Journal of Robotics Research*, 4(2):3–9, 1985.

[167] G. Zeng and A. Hemami. An overview of robot force control. *Robotica*, 15(5): 473–482, 1997.