High-Speed Robot Control in Complex Environments

By

Wyatt S. Newman

S.B., Harvard College (1978)
S.M., M.E. Massachusetts Institute of Technology (1980)
M.S.E.E. Columbia University (1982)

Submitted to the Department of Mechanical Engineering in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy
at the
Massachusetts Institute of Technology
October, 1987

Signature of Author [illegible]

Department of Mechanical Engineering
October, 1987

Certified by [illegible]

Prof. Neville Hogan
Thesis Supervisor

Accepted by [illegible]

Prof. A. A. Sonin
Chairman, Department Graduate Committee

High-Speed Robot Control in Complex Environments

by

Wyatt S. Newman

Submitted to the Department of Mechanical Engineering on October 21, 1987 in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Mechanical Engineering

ABSTRACT

Limits to high-speed robot control in complex environments are investigated both theoretically and experimentally. The work focuses on dynamically-decoupled robots, which have been shown to be fast and controllable. Three levels of control are investigated in theory and tested experimentally. The levels are: robust near time-optimal control, "reflex" control, and simple local planning. All three levels are integrated using configuration space. Experimental implementation is described for a two degree of freedom, planar, dynamically decoupled arm.

First, a nonlinear nearly time-optimal control technique is derived and shown to be globally stable and robust with respect to parametric uncertainty, unmodelled dynamics, and actuator saturation. Test results show an improvement in speed of a factor of four over conventional linear control.

Variations on potential function control are used to define "reflex" control, with which automatic obstacle avoidance is implemented in coordination with high-speed servo control. Configuration space is used to define a common representation for integrating high-speed motion with reflexes. Methods for computing 2-D configuration space transformations of moving obstacles at video rates (30Hz) are presented.

Reflex control is shown effective in preventing collisions, but does not guarantee effective progress toward a goal. A low-level local planning algorithm is described with variations for integration with reflexes and high-speed servoing.

Integration of the three levels of control leads to a suggested layered control organization in which cycle times are incrementally longer and complexity is incrementally broader for successive layers. To accomplish integration of multiple layers without deterioration of performance, each layer is defined as "expert" within some regime in which it assumes full control responsibility.

Thesis Supervisor and Chairman: Prof. Neville Hogan
Title: Associate Professor of Mechanical Engineering

Thesis Committee Members: Prof. Warren Seering
Prof. Jean-Jacques Slotine
Prof. Joseph L. Smith

To my family: Peggy, Guinevere, Edith, Warren, Sherry, Connie and Roxanne

Acknowledgement

This research was performed *in absentia* at Philips Laboratories, North American Philips Corporation, an arrangement which required patience, support and trust from a number of people both at M.I.T. and at Philips. I wish to thank my advisor, Neville Hogan, for his unflagging enthusiasm and encouragement. His energy and curiosity were a constant inspiration, and his personal efforts helped bridge the distance between Cambridge and Briarcliff Manor. Similarly, the rest of my committee, Warren Seering, Jean-Jacques Slotine, and Joseph L. Smith, all provided technical and moral support. My thanks to them for maintaining an active interest in my work, and for the effort they invested to follow it closely enough to offer sound advice and direction.

I owe more than my thanks to many people at Philips as well. Support for my work was continued through various changes in administration. Special thanks to Dr. Mark Rochkind, Alex Daniels, Dr. Arthur Sanderson and Dr. Ernest Kent for their support of my research. Many people at Philips Labs offered their technical advice in the construction of the experimental apparatus. My thanks to Arnold Eventoff for brainstorming and numerous discussions in mechanical design of the arm. For advice and assistance in electro-optics, I owe my thanks to Doug Stanton and Dr. Stewart McKechnie. Pat Maier contributed valuable assistance in assembly, characterization and initial control of the planar arm. Thanks also to Frank Guida for assistance in construction and numerous repairs to the robot. Two people in particular contributed their indispensable talents to the creation of the experimental apparatus: Erwin Klinger and Tom Harosia. Erwin machined most of the robot, working only from my illegible and incomplete hand sketches. In addition to independently catching my errors, he contributed novel ideas which were incorporated in the machine. Tom Harosia contributed electronic design, computer interfacing and device-level programming expertise. He stayed with me through two multiprocessing systems and three operating systems. Thanks Tom.

Finally, and most importantly, I want to thank my wife, Peggy Gallagher. Over the course of this project, she left the turmoil of her high-pressure job at work, and offered me her comfort and optimism while simultaneously responding to the somewhat less childish and selfish demands of our two year old.

TABLE OF CONTENTS

List of Figures

CHAPTER I

INTRODUCTION

Robots were invented, in concept, over 60 years ago, by the playwright Karel Capek in his play "Rossum's Universal Robots." In science fiction literature, the state of the art of robotics advanced by leaps and bounds. Meanwhile, the technological state of the art lagged by a disappointing margin. Some 30 years after Capek's introduction, the first robot patent application was filed. In 1961, the first industrial robot was introduced by Unimation. It took 16 years before Unimation showed its first profit. Meanwhile, more than 200 efforts worldwide had been launched in robotics, most of which were abandoned. While the virtues of robots were sung by robot manufacturers, many ambitious industrial projects in robot automated manufacturing resulted in disheartening failures.

Industrial robots seem to inspire a sense of familiarity in the human observer. Robot arms and grippers are typically suggestive of our own biological arms and hands. The eerie sense of familiarity prompts us to compare the robot's behavior to our own, a comparison which is inevitably a huge disappointment. In short, robots are slow, clumsy and stupid, making these iron-collar workers easily categorized as "unemployable."

Research described in this document was motivated by the author's curiosity as to why robots should be so incompetent. Examples abound of high-speed machinery in fixed automation systems which dramatically outperform humans at specific tasks. A wide selection of devices exist covering a variety, range and resolution of sensing well beyond the capabilities of human sensors. Powerful computing systems exist which can perform extremely fast logical and arithmetic evaluations. In the research presented here, it was the author's hope to uncover fundamental issues in the limits to high-performance robot systems.

A long-range objective of robotics research is the realization of a system which accepts commands as abstract goals, and accomplishes the goals effectively and efficiently. The robot should perform its own planning and respond automatically to unexpected changes. Such performance should be achieved without the need for detailed programming, as the robot system should handle all details autonomously.

In the present research, a more limited goal was defined: fast, safe and effective motion of a 2-degree-of-freedom robot arm servoing to a sensed, moving goal in cluttered, dynamic environments. The approach taken was "bottom-up", starting from the design and construction of a very fast robot arm, a high-bandwidth optical position sensor, and a powerful multiprocessor computing system. Control system development proceeded in layers, starting with robust time-optimal control. Control layers of "reflexes" and local planning were defined and integrated as concurrent processes. Techniques were developed and implemented for high-speed computations of obstacle shapes in terms of robot joint coordinates. Throughout, issues of generality were considered. Although the scope of the work included only motion execution, the control analysis and implementation were performed with the intent of making a system with a fundamentally sound foundation on which higher levels of abstraction in control could be built.

This thesis is organized in seven chapters. Chapters 2 through 6 present contributions which are logically distinct, but which integrate as a whole. Chapter 2 covers the design philosophy, design details and performance measurements of the three major subsystems (arm; sensing; computing) of the experimental apparatus. In Chapter 3, the lowest level control process is described: robust time-optimal control. Chapter 4 presents theoretical and algorithmic developments for executing fast transformations of sensed obstacles into equivalent forbidden regions in joint space. Reflex control is introduced in Chapter 5. Here, the author's first instance of control integration is described, in which execution of high-speed motion and guaranteed obstacle avoidance are realized simultaneously without deterioration of the performance of either control layer. In Chapter 6, integration issues are explored more fully. Following the suggested control organization guidelines, a third layer of control, local path planning, is introduced. Chapter 7 concludes with a summary and analysis of the work.

Each of the major ideas presented in Chapters 2 through 6 were treated theoretically and tested empirically. High-speed performance was a criterion for evaluating success throughout. As a heuristic benchmark, machine performance was considered adequate only if it outperformed a human by a substantial margin. The experimental hardware was used to verify theoretical contributions and demonstrate their effectiveness in machine control. More importantly, though, it was the author's impression that the machine directed the development of its own control structure. Theory could not be force-fit to the hardware in violation of improper, often unrecognized implicit assumptions. Organization and implementation of the control system was harshly judged by the machine, which seemed to demand its own preferred control design. On occasion, exasperated trial-and-error attempts led to unanticipated specific successes, which in turn generalized to broader theoretical contributions. In the author's opinion, metal and silicon introduced uncompromising judgment and truthful guidance which justified the bottom-up approach.

CHAPTER II

EXPERIMENTAL APPARATUS: DESIGN AND CHARACTERIZATION

§1. Introduction

A goal of the work described in this document has been the investigation of limits to high-speed operation of robots. To this end, an experimental apparatus was designed and constructed which intentionally simplified the experimental domain. The experimental apparatus consisted of three primary subsystems: an electromechanical robot arm, an optical sensing system, and a computing system. In the design of each subsystem, the priority was to emulate futuristic, high-performance machines and devices. Generality and direct industrial applicability were compromised in favor of constructing a system which would help spotlight fundamental issues in the limits to high-performance machine control.

Simplifications in the design of the robot arm included: limiting the mobility to two degrees of freedom in a plane; ignoring issues of grippers and end-effectors; ignoring issues of payload capacity and payload dynamic effects; and decoupling the dynamics through (pseudo) direct drive actuation and dynamic balancing. Since the arm was confined to motion in a plane, associated issues in sensing and in dynamic and kinematic computations were considerably simplified. Further, since the robot had only two moving links and did not carry an end effector or other payload, it was possible to design it to be light and stiff. Low inertia permitted high accelerations, and the high stiffness permitted high-bandwidth control while bypassing the complications of structural flexibility. Most importantly, the direct drive, dynamically decoupled design resulted in tremendous simplification of the dynamic equations, an effect which had strong influences on the structure and effectiveness of the control system design.

Simplifications were also introduced in the design of the sensing system. It was not within the scope of this investigation to consider issues in machine vision, e.g. high-speed image processing, object recognition, or stereo camera telemetry. Rather, some means of non-contact sensing was desired which would imitate an ideal non-contact sensor, presumably an advanced vision system. The chief simplification introduced was that each object to be sensed was instrumented with a "beacon": a set of infrared light emitting diodes (LED's). The sensor employed a planar photodiode which provided an analog indication of the x-y position of the centroid of all collected light. By flashing the light beacons individually, instrumented objects were unambiguously identified and located. Since the experimental domain was constrained to a plane, no range information was required. In this manner, the sensing problem was largely sidestepped; no real image processing was performed. The result, though, non-contact determination of object positions and identities, was the same as would be delivered by a (successful) vision system. However, the simplified optical sensor employed obtained this information two orders of magnitude faster than today's fastest vision systems.

The third subsystem, the computing system, was also intentionally overdesigned. In all, six powerful processors (MC68020) with a common high-speed backplane (VME) and 17 megabytes of shared random access memory were coupled in a multiprocessing system. Certainly, this level of computational capacity seems disproportionately high for the control of such a simple robot arm. Excess capacity was desired, however, so that limitations of computational power would not disguise more fundamental limitations in robot control. Further, what today is considered overkill in computational capacity will undoubtedly be considered modest in the near future. Like the robot design and the optical sensor, the multiprocessor emulates a futuristic version of currently available elements of robot systems.

In the following text, details of the three major subsystems are provided.

§2. Experimental Arm Design

In this section, the design philosophy, design details, and performance data are presented for the two degree-of-freedom, pseudo direct-drive dynamically decoupled planar robot arm.

§*2.1 Design Philosophy*

Design of high-performance robot arms has been investigated by a number of researchers. Some of the fastest experimental robot designs are those described in [2,3,4,5,13,21,42,52], in which direct-drive principles have been utilized.

The expression "direct drive" has become an increasingly popular phrase in machine design, though some confusion exists over what constitutes direct drive. A strict interpretation of direct drive is that no transmission is used; a moving link is rigidly attached directly to the moving element of the drive system (typically an electric motor). In spirit, a direct-drive machine is one which does not exhibit the nonlinear properties characteristic of transmissions. In fact, some robot designs are referred to as direct drive, although they employ some form of transmission. The M.I.T. "direct-drive" arm [3,4] actually directly drives only two of its three degrees of freedom; the third link is driven through a parallelogram linkage. In effect, though, the machine behaves like a direct-drive robot, since the linkage does not introduce significant friction, backlash or compliance. Similarly, the AdeptOne robot [13], is, in effect, direct drive in its first two links, although its second link is driven through a transmission. In the Adept design, the transmission consists of a pre-tensioned steel band between a drive pulley on the motor and a driven pulley on link 2. Backlash and friction of the steel band transmission are negligible.

One of the features of direct-drive robots which makes them fast is that the motors employed are markedly more powerful than those commonly used in industrial robots. A notable exception from this generalization of industrial robots is the AdeptOne robot which is essentially direct drive in its first two links, does utilize unusually powerful motors, and which is relatively fast. In a direct drive machine, the transmission is typically eliminated, and link torques are generated directly by the motor rotors, a fact which necessitates unusually powerful motors. Ordinarily, a transmission is used to amplify the relatively small torque developed by a drive motor. As a consequence, the speed required of the drive motor is correspondingly higher. A low power motor can not provide both high torque and high speed, so the overall performance of the system is limited by the capacity of the motors.

Actually, the use of more powerful motors permits a higher performance robot design, not necessarily through the elimination of a transmission. When better motors are used, a proper transmission design can still improve speed and acceleration capabilities beyond that of direct-drive [see, e.g., 36]. An optimized perfect transmission (zero backlash, zero friction, and zero compliance) would always be capable of improving a robot design over a transmissionless design. A direct-drive design, however, is easier to control than a machine which utilizes an imperfect transmission, since backlash, friction and compliance introduce nonlinear effects which are difficult to model, difficult to sense or infer, and difficult to stabilize within a control loop.

Another important trend in robot design is the use of remote actuation. In a design with remote actuation, the motor stators are secured to ground, as opposed to mounting them on successive moving links. In such designs, there is no penalty for the use of powerful, typically bulky motors, since the motor stator masses do not have to be accelerated along with the machine links. Strategic motor placement near ground is notable in the AdeptOne design, in the M.I.T. direct-drive arm design, as well as in the designs of the J.P.L. and Utah/M.I.T. hands [41,18].

When massive motors are used, if the design permits mounting the motors to ground then the links do not need to support the mass of the motors. Consequently, the links may be designed to be lightweight, yet have high resonant frequencies. High resonances are desirable, since it is recognized that low resonances effectively limit practical controller designs to low bandwidths. Attempts to control a robot at bandwidths beyond the system resonant frequencies are experimental at this point. Published techniques are difficult to implement and are very sensitive to modeling errors and parameter variations [9,12]. In practice, a robot design with high resonant frequencies is easier to control [16,51].

Remote actuation is apparently inconsistent with direct drive. However, the essence of direct drive, the elimination of the undesirable effects of common transmissions, can be realized although the actuators are placed remote from the respective driven links. With remote actuation, a particularly useful property can be achieved: decoupled actuation. With decoupled actuation, the actuators may exert efforts on individual links with respect to ground, without exerting direct reaction efforts on the remaining links. Dynamic coupling may still exist, e.g. from Coriolis or centrifugal forces, but there is no direct influence by the actuators on any but the intended links.

Figure 1 illustrates a hypothetical 2-D robot design which incorporates remote, decoupled actuation and virtual direct-drive behavior. With reference to this figure, actuation of one of the motors exerts a torque on only one of the links. The actuator for link 2 does have a transmission (a linkage), although it may essentially be ignored, since the transmission ratio is 1, and the efficiency of the linkage may approach unity. If motor 2 is driven, then there will be no reaction torque on link 1. Alternatively, if the motor for link 2 had been placed at the "elbow", any torque exerted on link 2 would be exerted equal and opposite on link 1. Decoupled actuation simplifies control, since feedforward decoupling torques do not have to be computed in software, and actuator saturation limits may be considered separately, regardless of simultaneous operation of multiple actuators.

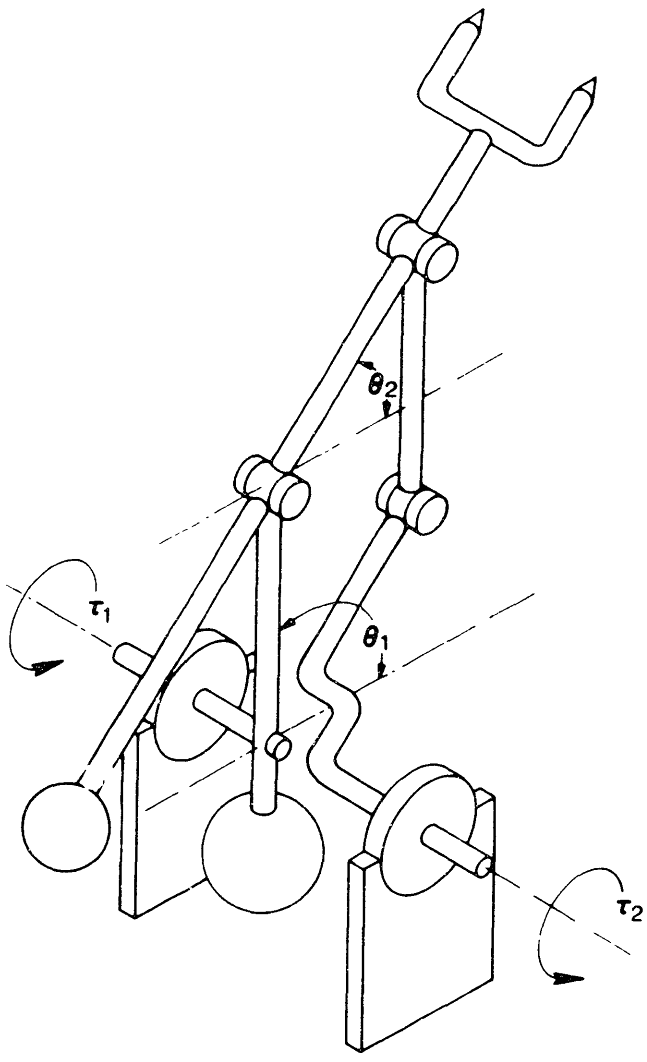

Figure 1: Two d.o.f. Dynamically Decoupled Robot Arm

An important additional development in robot design is the practice of dynamic decoupling. This technique was proposed by Youcef-Toumi in [52]. In order to achieve dynamic decoupling in a machine design with decoupled actuation, the mass distribution of each link is adjusted such that the center of gravity is in line with the respective axis of joint rotation. Dynamic decoupling has been designed into the M.I.T. direct-drive robot, at least for links 2 and 3.

Dynamic decoupling is illustrated in figure 1. Counterweights are shown on the backs of links 1 and 2, where the weight is selected such that the links are each neutrally stable under gravity loads. A consequence of such balancing, in combination with decoupled actuation, is that the dynamic equations are particularly simple in an appropriate reference frame. Following the convention of [6], the dynamics of a robot may be expressed in general as:

$$\sum_{j=1}^{n} H_{ij}\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} h_{ijk}\dot{q}_j\dot{q}_k + G_i = Q_i \tag{II.1}$$

where q_i is the i'th generalized coordinate, Q_i is the i'th generalized force, G_i is a gravity term influencing the i'th generalized displacement, and h_{ijk} are factors derived from spatial derivatives elements of the inertia tensor, H_{ij}.

For a dynamically decoupled manipulator, gravity terms are cancelled by balancing, and the inertia tensor is invariant. Since the inertia tensor is invariant, (inertias do not change as a function of the position of the robot), the terms h_{ijk}, which lead to Coriolis and centrifugal forces, are eliminated. Further, for the proper choice of generalized coordinates, the inertia tensor is diagonal. In particular, if absolute joint angles (i.e., angles referenced to ground rather than referenced to other links) are chosen as generalized coordinates, then the inertia tensor is diagonal. If, in addition, joint and transmission friction is low, i.e. the machine is effectively direct drive, then the robot dynamics degenerates to a decoupled collection of second-order, linear systems. Under these circumstances, the system dynamics may be written in standard linear state-space form as:

$$\dot{\vec{x}} = A\vec{x} + B\vec{\tau} \tag{II.2}$$

In the above, $\vec{x}$ is the state of the system in terms of joint positions and velocities, $\vec{\tau}$ is a vector of joint efforts, and A and B are time-invariant matrices. For a balanced manipulator, the A matrix can be arranged in block-diagonal form, and for decoupled actuation, the B matrix will be in control canonical form [defined, e.g. in 50]. The dynamic equations then describe a parallel array of reduced-order subsystems, where each actuator effort affects a corresponding subsystem independently.

Dynamic decoupling results in a remarkable simplification of a robot's control system. In fact, the simplification is so dramatic that dynamic control problems which are intractable for common manipulators are trivial for the dynamically decoupled manipulator [34].

Lessons from robot design studies have been incorporated in the design of the two degree-of-freedom planar arm used in the present investigation. The experimental arm utilizes oversized motors, remote actuation, virtual direct drive via a nearly ideal transmission, and counterbalancing for dynamic decoupling. Remote actuation permitted the design of lightweight, stiff links which achieved high resonant frequencies. Further, the oversized motors permitted unusually high link accelerations. The simplifications introduced by a nearly frictionless steel-band transmission and counterweights for dynamic decoupling resulted in a system which was particularly well suited for high-performance control.

§2.2 *Design Parameters*

Mechanical details of the experimental arm are shown in the photograph of figure 2 and in the sketch of figure 3. The arm consists of two links driven by respective d.c. servo motors through steel bands.

Link one consists of an aluminum box-beam, 2"x3" with a nominal wall thickness of 1/8". The beam was cut to a length of 12.5", and the wall thickness was machined down to 1/16" over most of the length. After machining, link one had a mass of 385 grams. Link one is driven by an aluminum torque-tube with a nominal diameter of 2" and a nominal wall thickness of 3/16". At the distal end of link 1, a recessed channel was milled for guiding upper and lower bearing plates, which housed ball bearings for a pulley secured to link 2.

Link 1 encloses a steel band which drives link 2. The steel band is type 301 high yield stainless steel, 1" wide, 0.004" thick and 26" long, laser welded along a diagonal seam to form a continuous loop. The drive pulley for link 2 is apparent in both figure 2 and 3. The drive pulley is a 2" diameter aluminum cylinder,

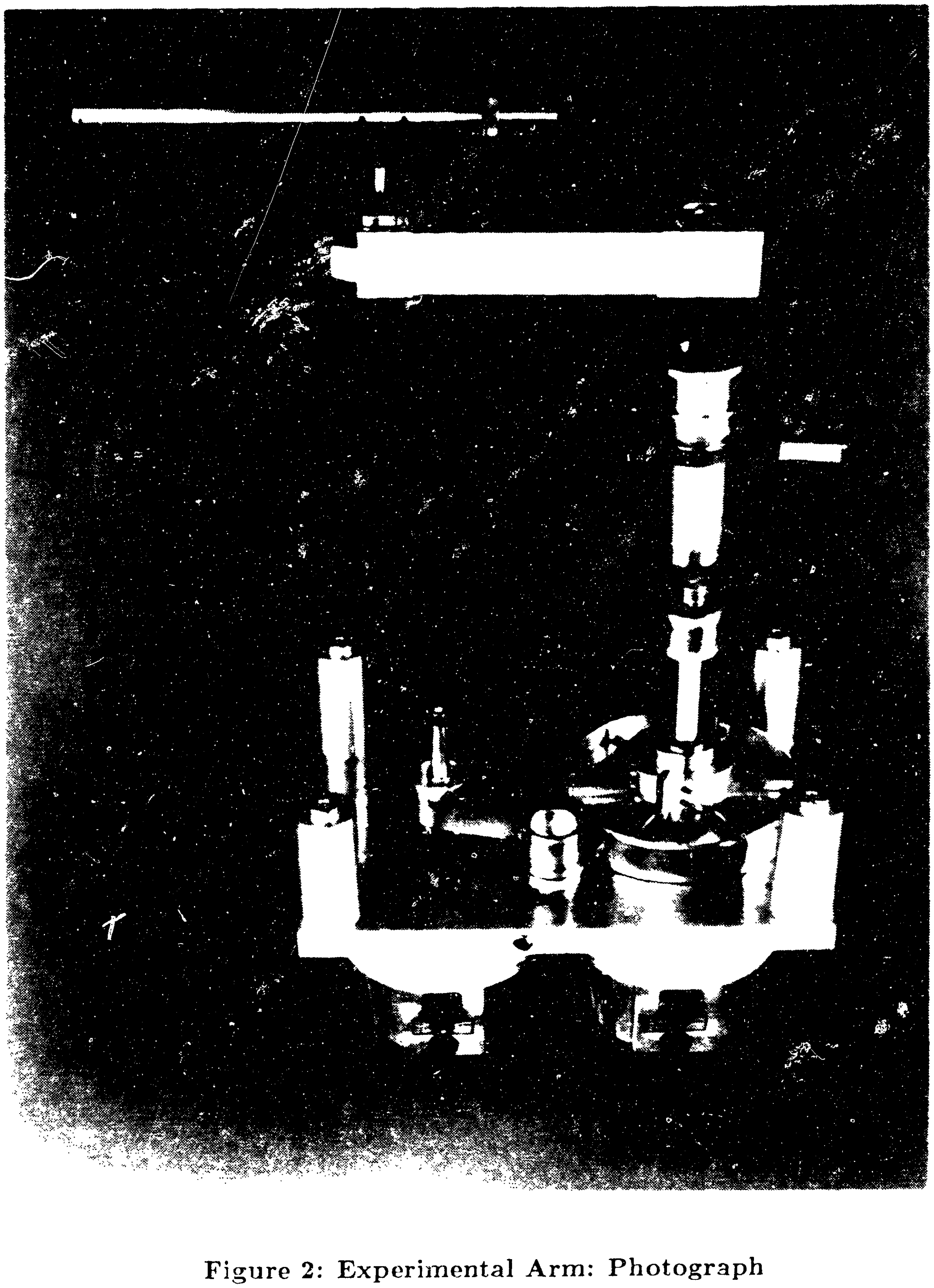

Figure 2: Experimental Arm: Photograph

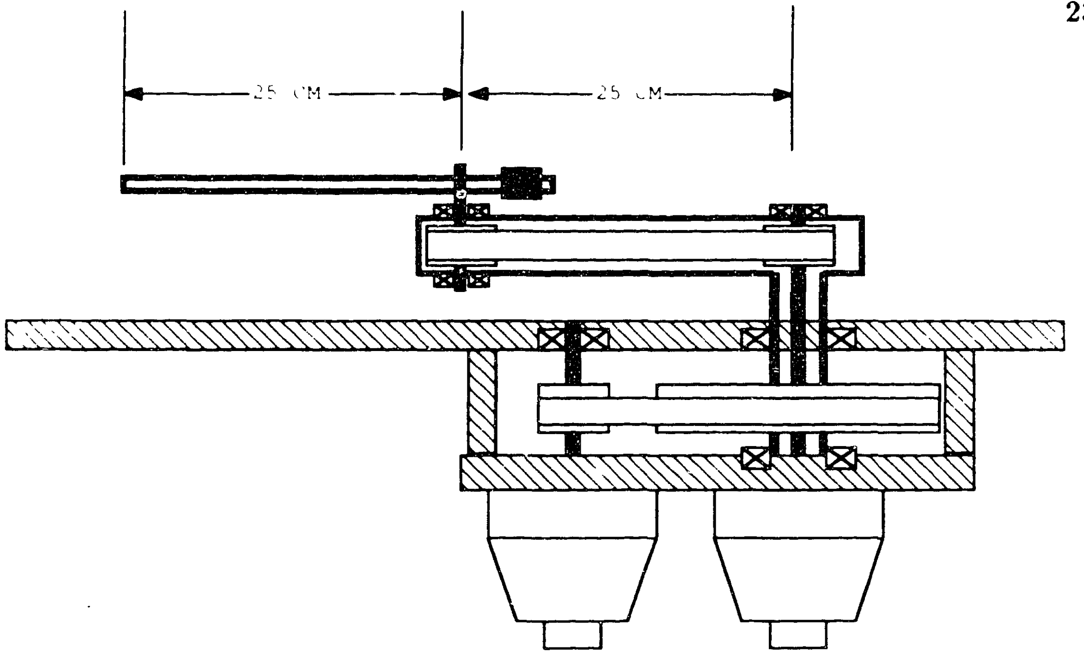

Figure 3: Experimental Arm: Layout

1.5" in length with a 1/2 degree crown to aid in belt centering. It is attached to a 7/8" diameter aluminum shaft, which is in turn joined to the shaft of motor 2 with a split-sleeve coupling. The drive shaft for link 2 is concentric with the torque tube which drives link 1.

Design of the transmissions was a critical aspect of the arm construction. Requirements were that the drive train be lightweight, yet exhibit high stiffness with zero backlash and friction. Traction drive steel bands met the requirements. For high stiffness, the cross-sectional area of the bands needed to be sufficiently large, though the thickness of the bands was limited by bending stresses induced by wrapping around the pulleys. Since the driven pulley for link 2 had to be supported at the end of link 1, the mass of that pulley had a significant influence on the inertia of link 1. Thus, the pulley mass was minimized by minimizing its dimensions, and fabricating it from magnesium. A pulley diameter of 2" was selected, resulting in a distal pulley mass of 130 grams. At the specified 1" radius of curvature, a belt with thickness of 0.004" undergoes a strain of

2000 microstrain in wrapping around the pulley. Relative to the bending strains, the loads induced by pretensioning and supporting the maximum motor output torque are negligible. At the chosen belt thickness to pulley diameter ratio, the expected fatigue life of the belt is roughly one million cycles. Over 14 months of inadvertently abusive experiments, no failures were encountered.

To prevent belt slippage, the belts were pretensioned to 45 lbf. At this pretension, the belts were always in tension, even at maximum motor torque. Further, the coefficient of friction between the steel belts and aluminum pulleys (measured at 0.61 static, 0.47 sliding) guaranteed no slip[1] at pretensions well below the chosen value. In order to pretension the steel bands, adjustment mechanisms were built into the design. For link 2, the bearing plates located on link 1 were designed to slide in channels along link 1. Adjustment screws on link 1 were provided to incrementally displace the bearing plates radially outward from the axis of link 1, which permitted stretching the link 2 drive belt slightly for pretensioning. The nominal distance between the drive and driven pulleys of link 2 was 25 cm.

Both the drive and driven pulleys for link 2 were 2" in diameter, providing a unity transmission ratio. The measured inertia of the link 2 system was 0.0035 Kg-m^2, of which the motor rotor inertia was only 17%. The optimal transmission ratio for the motor/load combination would have been 2.2:1, not 1:1. Use of the optimal transmission would have resulted in 32% higher acceleration capacity. However, a 1" radius pulley was the maximum tolerable curvature for long life of the 0.004" belt, so a transmission ratio of 2.2 would have required the use of a 4.4" diameter driven pulley. The use of such a large pulley would have carried a severe inertia penalty for link 1, which is already the slower of the two links. Thus the 1:1 transmission ratio was accepted as preferable.

Link 2 had a substantially larger inertia than link 1, so a speed-reducing transmission was desired. The chosen pulley ration was 4:1, realized with a 2" diameter pulley and a 8" diameter pulley. As can be seen in the photograph

[1]see, though, section 2.4 regarding creep

of figure 2, the driven (aluminum) pulley for link 1 consists of a thin rim, six spokes for lateral bending stiffness, and a thin circular plate joining the inner and outer radii for rotational stiffness. The resulting pulley is light and stiff, with a relatively low inertia. This pulley joins to the link 1 torque tube through a split collar. The link 1 pulleys also utilize a slight crown to help keep the steel belt centered. The belt for link 1 is identical to that of link 2, except for its circumference of 32.7". In figure 2, a third pulley can be seen in contact with the drive belt of link 1. This pulley is an idler with an adjustable location along a slot, which enables pretensioning of belt 1. Like belt 2, belt 1 is pretensioned to at least 45 lbf, though in practice the pretension has exceeded this value safely, since the bending stresses induced by wrapping around the 2" pulley dominate the stresses in the belt.

The inertia of link 1, including the driven pulley, the box beam, and the load incurred by supporting link 2 and its driven pulley, was measured via dynamic tests to be 0.037 Kg-m^2, or nearly 60 times the motor inertia. Such radical load imbalances are characteristic of direct-drive systems. With the chosen transmission ratio of 4:1, link 1 peak acceleration is improved by a factor of about 3.2. The optimal transmission ratio would have been about 8:1. Use of the optimal transmission would have theoretically improved the peak acceleration by an additional 25%. However, a ratio of 8:1 would have required an impractically large diameter driven pulley, which would have increased the inertia of link 1 significantly, thus reducing the peak acceleration. The chosen design is nearly optimal.

Link 2 is constructed from a simple tube of aluminum, 3/4" in diameter, 12.8" long, with a wall thickness of 0.040". The tube joins to its drive shaft, a 3/8" hollow stainless steel tube with a 1/16" wall thickness, using a magnesium clamp. The drive shaft is joined to the magnesium link 2 driven pulley by means of a thermal shrink fit. Link 2 weighs 80 grams, and the magnesium shaft clamp weighs 8 grams. In addition, link 2 supports an instrumented Delrin tip (12 grams) and brass counterweights (75 grams). The length of link 2 between its joint axis and the center of its tip is 25 cm. The counterweights are an important feature; they are used to shift the center of mass of link 2 to a position coincident with the joint 2 axis. The position of the weights is adjustable along the rear of link 2 by means of a fine-pitch thread. Two weights are rotated differentially to lock them in place at the determined balance position.

In order to get power or signals to or from the tip of link 2, slip rings have been provided. The slip rings and brushes are apparent on the torque tube of link 1 in figure 2. Slip rings and brushes are also located on the shaft of the distal link 2 pulley, though they are less obvious in the figure. The slip rings consist of grooves machined in a phenolic ring, inlaid with 0.005" x 0.020" conductive metal strip. The brushes are 0.012" diameter wires, which are pretensioned against the conductive strips. Eight parallel tracks were installed for link 2, and 4 for link 1. Redundant brush packs were installed and wired in parallel to reduce brush noise. The conductive strip material was Paliney 7, and the wires were Neyoro G, both proprietary alloys of precious metal (J.M. Ney Co., Bloomfield Conn.)-optimized for long-life electrical brush applications.

The motors selected to drive links 1 and 2 are pancake-type printed circuit armature d.c. servo motors (JR16M4CH, PMI Motion Technologies, Commack, NY). The motors have a particularly low inertia and inductance, which makes them well-suited for high acceleration servo applications. Since the links are remotely actuated (by means of the steel bands), the motors are mounted to ground. Thus, their mass, which is disproportionate to the mass of the links, does not need to move, and the design is not penalized for using oversized motors.

No provision was made in the system design for forced-air cooling of the motors. The maximum continuous current rating without forced cooling is 9.6 amps, at which the motors produce a safe, continuous stall torque of 3.5 N-m (495 oz-in). Peak torque is limited by demagnetization of the permanent magnets and by transient heating; the rated peak torque is an order of magnitude larger than the stall torque. In practice, the transient torque capability of the motors was not utilized. The amplifiers were set to current limit at the rated continuous stall current of the motors. This was done to protect the motors, since a low duty cycle of peak torque commands could not be assured during normal operation.

The motors were driven by pulse-width modulated amplifiers (PWM) operated in transconductance mode. The amplifiers (PMI CX-90/125) operated at a switching frequency of 6kHz with voltage limits of +/- 125 Volts and current limits of 30 Amps peak, 15 Amps continuous. The continuous power output capacity was 1.875 kW per amplifier. In operation, the amplifiers were set to current limit at about 10 Amps, the maximum safe continuous current of the motors. The amplifier circuitry was modified by the author for operation in transconductance mode; the gain was adjusted to 1 Amp motor current per 1 Volt input command.

Optical incremental position encoders and tachometer/generators were purchased installed on the motors for link position and velocity sensing. The optical encoders output A and B pulse trains in quadrature at 2048 pulses per revolution. Utilizing all four transitions of the A and B signals per pulse, rotor position was detected to one part in 8192, for a resolution of 0.77 milli-radians. For link 1, with a pulley reduction of 4:1, the angular resolution of link 1 motion was 1 part in 32768: about 0.19 milli-radians, or 40 arc-seconds. The tachometer sensitivity was 3 Volts/ 1000 rpm. Although the robot was capable of high accelerations, its velocity was never large for any normal move (less than a full rotation). As a result, the tachometer sensitivity was too low to be of use. At low velocities, noise generated by the switching amplifiers dominated the small signal generated by the tachs.

§*2.3* *Performance Measurements*

The success or failure of the design can be characterized by four crucial measurements: the bandwidth of the amplifiers; the resonant frequencies of the links; the maximum acceleration of the links; and Coulomb friction.

Amplifier bandwidth was measured by obtaining a transfer function of the current output vs the command voltage with the motors connected. White noise

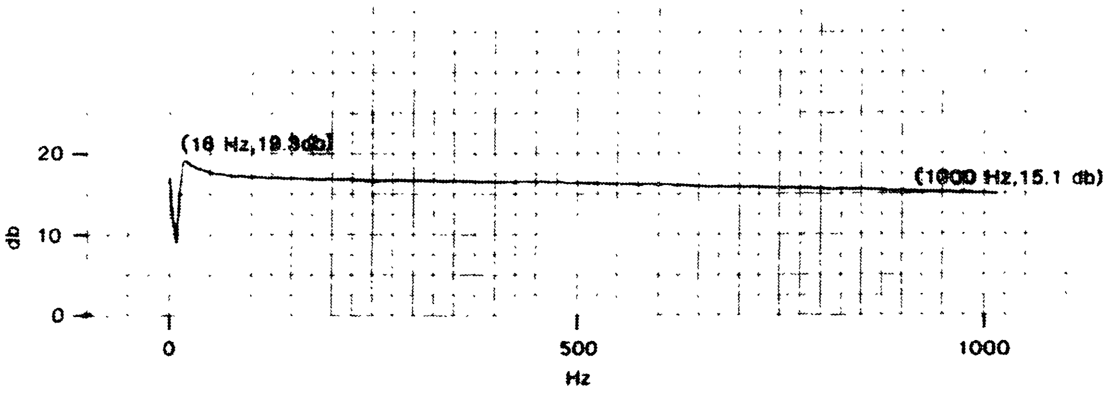

Figure 4: Transconductance Amplifier Frequency Response

was sent to the amplifier input, and motor current was measured and processed by a spectrum analyzer. During these tests, the motor rotors were not locked, since, in transconductance mode, the amplifiers were controlled to compensate for motor armature resistance, inductance and back EMF. A sample plot of the results appears in figure 4, which shows output current vs input voltage as a function of frequency. Due to the nature of the FFT algorithm, data near 0 frequency is not valid. The response is actually quite flat from 0 Hz out to about 1.5kHz. An inconvenient artifact of the spectrum analyzer is that the frequency scale is linear rather than logarithmic. Thus, it is difficult to infer a system model by inspection of the transfer function plot. Nonetheless, it is clear that the amplifiers are essentially ideal over a wide range.

Transfer function tests were performed on the links as well. An accelerometer was mounted to link 1 near joint 2, and white noise was sent to the amplifier input of link 1. The resulting acceleration was processed by a digital spectrum analyzer. The same test was repeated for link 2, with the accelerometer mounted on the tip of link 2. The results are shown in figures 5 and 6, respectively.

Link 1 tests revealed a first resonance at 212 Hz. The first link 2 resonance appeared at 125 Hz. The tests are most accurate for link 1, since the added mass of the accelerometer (13 gm) hardly influenced the load of link 1, but did

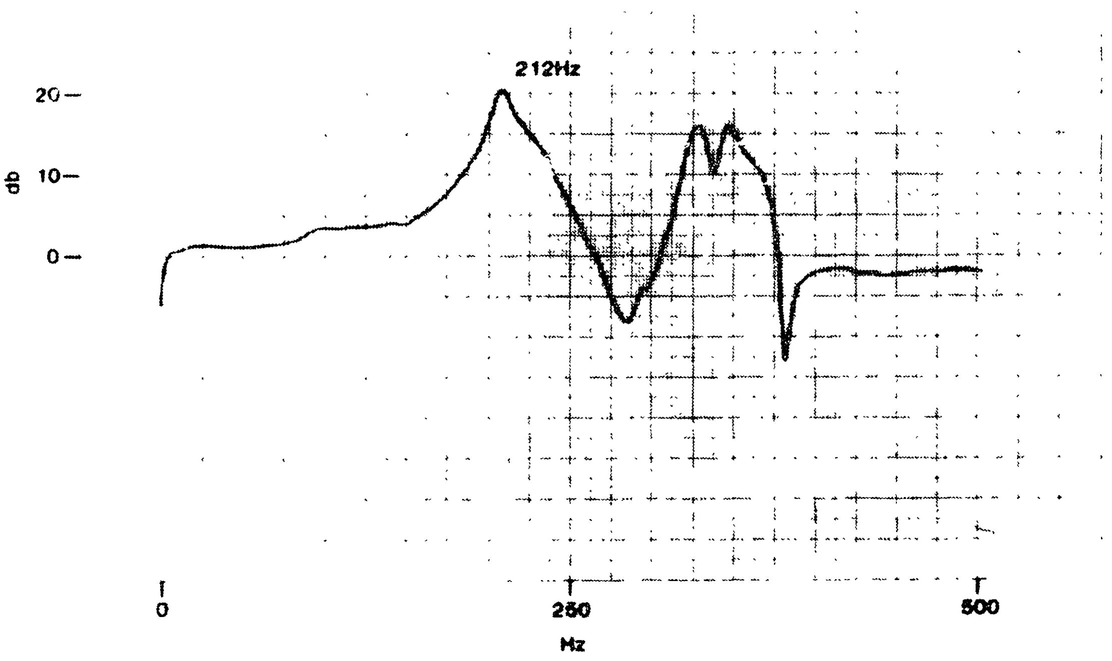

Figure 5: Link 1 Acceleration/Torque vs Frequency

have a nonnegligible effect on link 2. With the added mass of the accelerometer, the observed resonant frequency of link 2 is a conservative underestimate of the unloaded natural frequency. Nonetheless, the measured frequencies indicate an unusually stiff system.

Peak acceleration measurements were obtained both with an accelerometer, and by step-input torque transients with rapid position sampling of the encoders. Measured accelerations at the maximum safe continuous current lev .s were 290 r/s^2 and 960 r/s^2 for links 1 and 2, respectively.

Friction was measured using a force gauge to back-drive the links with the amplifiers off. Measured frictional torques were 0.30 N-m for link 1 and 0.085 N-m for link 2. The measured friction is attributable entirely to friction inherent in the motor armature, presumably due to the brushes.

Overall, the electro-mechanical system met the design objectives of providing an extremely fast and controllable arm.

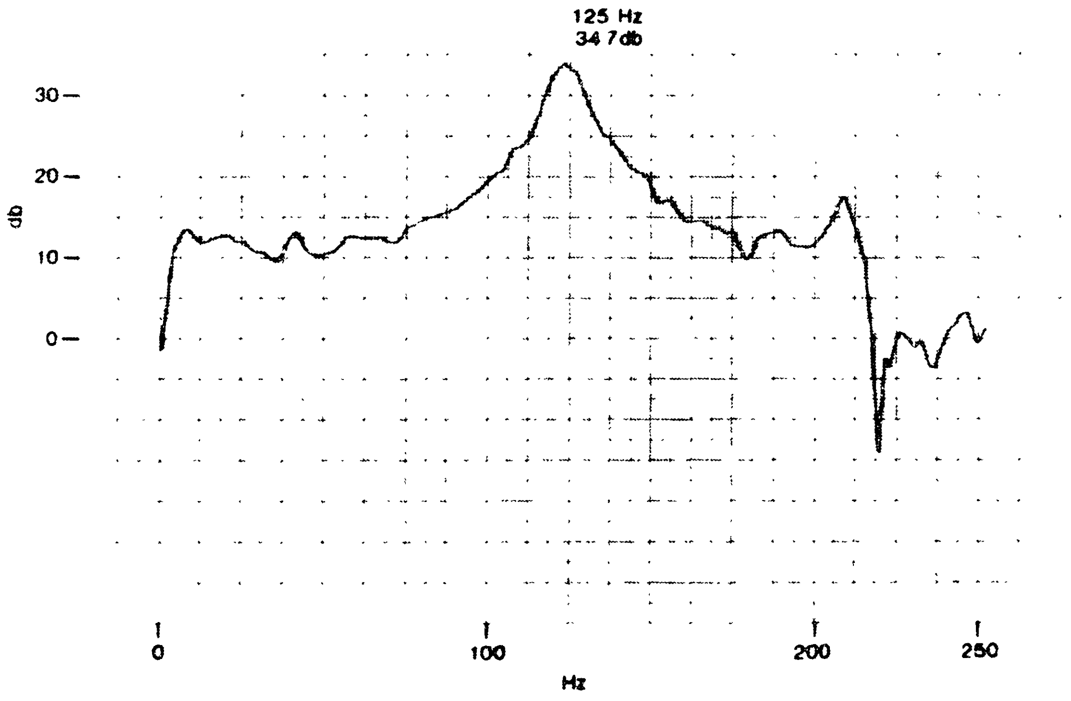

Figure 6: Link 2 Acceleration/Torque vs Frequency

§2.4 *Suggested Modifications*

In using the experimental apparatus, it became apparent that the sensing means could be improved. As mentioned earlier, the tachometers did not turn out to be useful, since their signals were too small at low velocities. Velocities inferred from t.:e incremental encoders were reasonably accurate at moderate velocities, but problems were encountered at low velocities. At low speeds, the data rate from the optical encoders is too slow to specify the velocity. Between pulses from the encoders, it is not clear how the velocity might be changing. Better velocity sensing is desirable. The author suggests the use of resolvers in place of the optical encoders, since velocity information is directly available from the resolver converter electronics.

A second sensing problem is that the optical enco [illegible] are located directly on the motors rather than on the links. Thus, information about link positions must be inferred from the motor rotor positions. However, since the machine is capable of continuous rotation of both links, nonideal (non-integer) transmission ratios introduce errors which can accumulate to the point where the motor rotor angle is a poor estimate of the link angle. It would be better to place the sensors directly on the links, rather than on the motors. At the least, some type of home sensor, independent of the motors, should be implemented to null accumulating errors.

Another source of errors between motor position and link position is the phenomenon of belt creep. Although the belt does not slip if adequately pretensioned, belt creep can occur under high accelerations. When the belt transmits a large torque, one side of the belt is in greater tension than the other. Consequently, as the pulley rotates, the section of the belt which rolls onto the pulley is under a different strain than the section which rolls off; stretched (elongated) belt winds onto the drive pulley while relaxed belt winds off. As a result, the belt seems to "worm" its way across the pulley. This effect does not restrict the controllability of the system, but it does invalidate the placement of sensors on the motors for high precision position sensing.

§3. Optical Sensing System

In this section, the optical sensing system is described, beginning with an overview of the design concept followed by design specifics and performance data.

§*3.1 Design Concept*

As mentioned in the introduction, no attempt has been made to implement a general vision system. Existing vision systems are expensive, complex, unreliable and slow. In a typical vision system, a charge coupled device (CCD) detector is used in the image plane of a camera lens. The CCD detector samples light intensities at discrete "pixels" in a square array, most commonly 256x256 elements. The intensity values of the individual elements are read out serially, typically requiring 16.7 msec to transfer the complete set of data. An image processor then analyzes the data, either statistically or by extracting low-level image features such as edges and corners, to try to identify and locate objects in the field of view of the sensor. Image analysis is a computationally intensive process, which can take hours on mini computers. One of the fastest specialized image processors to date, the PIPE machine [19], can extract low-level features in "real-time". Object identification still requires additional, slower processing. In the area of machine vision, real-time processing is a highly ambitious goal, though, "real-time" in this field generally implies a mere 60 Hz or lower throughput, and substantial pipeline processing delays from input to output. Such limitations makes vision currently unsuitable for use in high-performance feedback systems.

Vision systems of the future may be expected to overcome the current severe restrictions. For the present experiments, a detector was desired which would emulate a futuristic vision system. Thus, it was decided to simplify the problem domain by marking each obstacle to be detected with a light beacon. Further, rather than tolerate the time delay associated with serial communication of discrete data, an analog detector was used.

The key component of the sensing system design was a PIN (P-type layer, Intrinsic layer, N-type layer) lateral effect photodiode. This two-axis detector has four electrodes on its edges. When a light spot is focused on the sensitive area, a photocurrent is produced which divides up among the four electrodes, roughly proportional to the distance of the light spot from those electrodes. Thus, by measuring the currents through each of the electrodes, the position of the light spot can be inferred. In this manner, generality (recognition of arbitrary objects) has been sacrificed for speed. Whereas 60 Hz is considered exceptionally fast for general vision systems, PIN photodiodes can respond at gigahertz rates.

With a planar photodiode detector, it is not necessary to sharply focus light inputs. Since the detector is roughly linear in spot position vs electrode current, a blurred spot results in the same electrode currents as a focused spot. The electrode currents divide up according to the position of the centroid of the light hitting the detector. If a single light source is observed by the detector, then the detector currents determine the position of that source. The light source need not be a point source, though; the sensed position of a distributed source is the centroid of the light emitters. Therefore, light power emitted by a marked object, and thus signal strength of the detector, may be increased by employing multiple emitters per object. By the same process, however, if light beacons were energized simultaneously on distinct objects, the signal would be perceived as a single input, and a meaningless conglomerate position measurement would result.

In order to detect multiple objects, the light beacons of each object are flashed sequentially, one light source at a time. Thus, multiple targets are perceived by multiplexing their signals, and associating the detector currents with the light flashed at any instant. Consequently, the sample rate of detecting any one object is inversely proportional to the number of objects. In the present design, eight objects are detected.

Four signals (the currents through each of the electrodes) are provided by the detecting element, although only two pieces of information are desired: the x and y positions of a light spot. A pair of electrodes essentially determines the light spot position along each axis, where the current difference of each pair is (nearly) proportional to the spot position along the respective axis. As the intensity of the spot increases, the currents of each electrode increase nearly linearly. Therefore, an absolute position measurement can be obtained by normalizing the difference in currents for each electrode pair with respect to the sum of the respective currents. The resulting signals are relatively insensitive to varying intensities of the target light sources. In the present design, the normalization process is done in analog. Currents through each electrode are sensed, and the difference and sum of each pair are obtained as analog voltages. The sum and difference voltages for each pair are in turn used as numerator and denominator inputs, respectively, to analog division chips. The result is a pair of analog voltages which are a measure of the x and y positions of a detected light source.

Coordination of flashing light beacons and acquisition of sensor data is performed by a control computer. The computer flashes the LED of each object for a programmed length of time, samples the detector voltages, performs digital filtering of the position signals, and stores the result in memory where it is accessible to other processors. The sensing process is cycled continuously, and the most recent position information for each obstacle is always made available in shared memory. In this manner, the use of position information is made independent of the specifics of the sensor. Any type of sensor at all may replace the chosen electro-optical system without affecting the control code running on other processors. In particular, an advanced vision system could replace the current optical detector; the only requirement is that the results be made available in common memory.

§3.2 *Electro-Optical Design*

The heart of the position sensing system is a lateral effect photodiode, (PIN-SC25D, United Detector Technology, Hawthorne, CA). The detector has a square sensitive area, 0.74"X0.74". The responsivity is 0.25 Amps electrode current per Watt of light power input over a spectral range of 350nm to 1100nm. In order to minimize interference from room lighting, a daylight filter (850 nm half power point) was installed between the lens and detector, which blocked visible light and passed near infrared (865 nm).

Electrode currents were sensed, amplified, and input to an analog divider chip (Burr-Brown DIV 100 HP). The analog divider chip had a small-signal frequency response of 350 kHz, and a full-power bandwidth of 30kHz. The resulting signals were filtered with first order active filters with a 40 microsecond time constant, and input to parallel 12-bit analog to digital converters (AD574A, Analog Devices, Norwood, Mass.).

Infrared light-emitting diodes were used as light sources. Three diodes (TRW Optron OP233W) were wired in series for each object. Instrumented objects included five cylindrical Delrin obstacles, the tip of the robot, the tip of a hand-held wand, and the wrist of a small stepper-motor driven instructional robot (Microbot Teachmover, Microbot, Inc, Mountain View, CA). Pulse currents of 560 mA were excited sequentially in each of the 8 sets of diodes, resulting in a 12.5% duty cycle. The rated continuous current for the LED's is 100 mA, at which the power dissipation is 200 mW. At 560mA, the power dissipation is 8 times greater, so a duty cycle of 12.5% results in an LED life corresponding to continuous operation at rated continuous current. Under continuous operation, the light output will slowly degrade (about 5% over 1000 hours). The wavelength of emission is 880 nm, at which the energy is hardly attenuated by the daylight filter. Power output is 45mW per LED at 560 mA current, which is distributed at nearly constant intensity +/- 20 degrees about normal, with sharp attenuation at greater angles.

The detector was placed behind a wide-angle lens (28mm focal length), with a 19mm aperture. The lens collected and focused light energy over a 32mm diameter. The system was placed 1.6m above the plane of the LED's, resulting in a field of view 1 meter in diameter; since the detector was square, the actual sensed area was square, 0.7x0.7m (1 meter diagonal). At 1.6 meters above the plane of the light sources, the wide-angle lens intercepted only 0.05% of the emitted light power, or about 70 microWatts of light power.

Each LED was pulsed for a duration of 250 microseconds. The control computer initiated each current pulse, then waited the full pulse duration before sampling the analog detector signals. A pulse period of 250 microseconds allowed adequate time for the the analog dividers and the analog filters to settle. For the eight marked objects, the resulting sample rate was 500 Hz for each object.

§3.3 *Sensory Control Computer*

A single-board computer was used to control the timing and sequence of beacon flashes, to sample the detector data, and to perform additional filtering and communicate the results to additional computers. For the requirements of the sensing system, a simple processor or discrete logic would have been adequate. However, for compatibility with the rest of the parallel processing system and for ease of development and modifications, a relatively sophisticated computer used which was identical to the other processors in the computing system. The sensory computer (PV682, Pacific Microcomputers, Inc, Cardiff, CA), was based on the Motorola MC68020 microprocessor running at 16.7 MHz. The computer board included 1 Mbyte of RAM, and was VME compatible. It was installed in a VME rack with a common backplane connection to the other five processors of the computing system. A common memory board, independent of all of the processor boards, was also installed in the VME rack. This memory was used for communications among the processors and, in particular, held the table of object positions which was continuously updated by the sensory computer. The position of each of the eight instrumented objects was distinguished, though the sensory computer assigned no information as to the use of the various positions. Ordinarily, the wand tip was treated as a goal position, and the cylinders were treated as obstacles. Such decisions,though, were not within the realm of the sensory processor.

A useful feature of the chosen computer boards was the addition of a private connector, independent of the VME bus, which was directly accessible by the CPU. The private connector was used as an interface port for custom electronics. For the sensory computer, a parallel input/output interface board was constructed for controlling the current drivers for the LED beacons, for initializing conversion cycles for the parallel analog to digital converters, for testing flags indicating completion of conversion cycles, and for reading the digital data from the converters. All of this I/O was performed by the sensory computer without

VME bus accesses. In this manner, bandwidth of the VME bus was reserved for communications among processors rather than private I/O.

In addition to coordinating the timing of the electro-optical system, the sensory computer also performed digital filtering of the input signals. Each of the 16 signals (x and y for 8 LED's) were filtered using a simple finite impulse response algorithm: a running average of the 32 most recent signals was continuously updated. The filtered x and y position signals were continuously updated in a table in shared memory at 500 Hz per object.

Both the detector and the wide-angle lens introduced distortions in the linearity of the position signals. Linearity was improved in operation by creating a calibration lookup table which converted raw (distorted) position information into approximate true coordinates. Calibration data was obtained by controlling the arm to servo to 1296 points in view of the sensor, spaced every 2 cm in a square grid, and flashing a set of LED's on the tip of the arm at each specified location. Actual LED position was computed from the link lengths and link angles (as indicated by the encoders) of the robot. The data gathered in this way was stored as voltages x and y vs actual x and y, i.e. two tabulated functions of two variables. The two functions were then inverted numerically to obtain tabulated functions of actual x and y vs signal voltages x and y.

§3.4 *Sensing System Performance*

The sensing system performed adequately, though not as well as was hoped. Noise and thermal drift limited the absolute accuracy and the resolution of the detected positions. After calibration and tabulation of position lookup tables, the position of LED's on the tip of the robot were optically sensed and compared with the computed tip position based on the encoder angles. After digital filtering (32 sample running average), the noise of the position signals corresponded to a position uncertainty of +/- 3mm over a range of 0.70 meters. This resolution limit corresponds to about 0.5%, approximately the same resolution as a 256x256

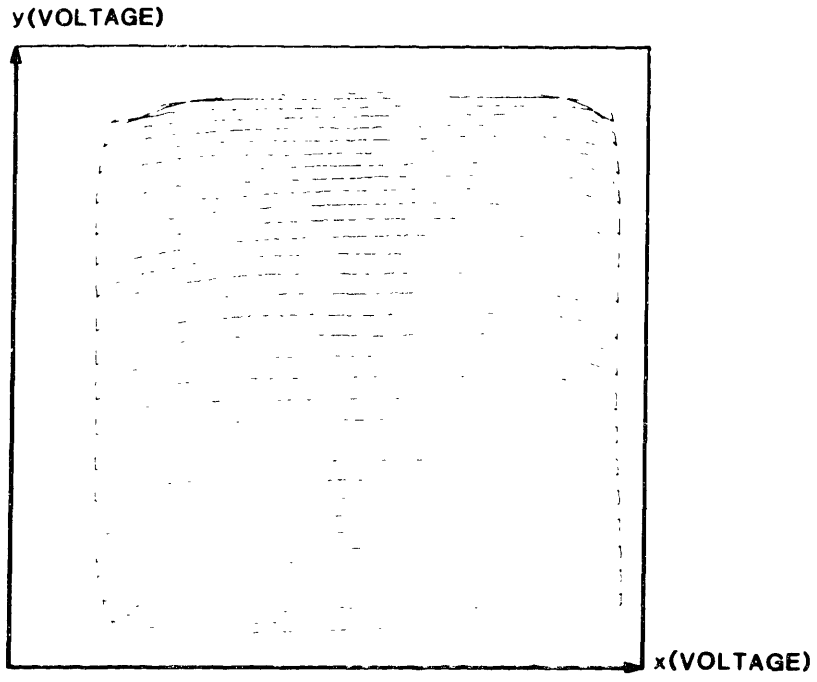

Figure 7: Sample Camera Calibration Scan

CCD detector. Thermal drift, however, resulted in steady offset errors as large as 18mm. For accurate sensing, it was necessary to allow the electronics to operate continuously for several hours to thermally equilibrate before performing a calibration scan. Sensing performed immediately after a recalibration did not suffer from a thermal offset.

Data from a sample calibration run is shown in figure 7. The figure shows x voltage vs y voltage of the detector sensing the robot tip as the robot was scanned back and forth over the field of view (0.70m by 0.70m). The path of the tip was horizontal (parallel to the x-axis) along each traversal between maximum and minimum x positions. Each line scan was displaced by 2cm in y. Thus, a perfect sensor would have resulted in a calibration plot with perfectly straight lines and a uniform separation between lines. In the actual calibration scan, it is apparent that there is significant distortion away from the center of view. In fact, the top-most line scan resulted in signals which overlapped the previous line scan, indicating that the robot tip was nearly out of view, and the data was distorted by lower intensity and possible reflections inside the lens and aperture arrangement. Near the center, the lines are nearly straight and have a relatively wide spacing. The spacing between lines is widest where the sensitivity is best, since a large spacing corresponds to a large voltage change, while the physical y-displacement of the tip between any two successive line scans was always 2cm.

A more serious problem encountered was that the position signals remained sensitive to LED intensities, in spite of normalization by analog division. Thus, not only thermal drift, but also LED aging resulted in misperception of LED locations. For the tip, periodic recalibration was sufficient to compensate for LED aging. For LED sets which marked obstacles, however, the intensities were not the same as that of the tip, resulting in position sensing errors. The LED current driver was not well regulated, and differences in LED currents were observed among the various LED's. With normalization, the minor differences in LED currents should not have mattered. In practice, x and y position errors as large as 15mm resulted from the LED intensity differences.

Background lighting did not present a problem, at least within the laboratory setting of the apparatus. Lighting conditions were essentially constant, and constant background effects were automatically compensated in the calibration process. Further, the laboratory lighting was fluorescent and thus had a low infrared output. The remaining spectral emission of the fluorescent lights was effectively blocked by the daylight filter. Also, since the infrared LED's confined virtually all of their light emission to a solid angle of 0.38 steradian (0.06% of a hemisphere), and since the emitters were all directed straight up at the overhead camera, transient reflections were not a problem. Moving objects were not permitted in the space illuminated by the LED's. Static objects within the illuminated space may have produced reflections which affected the camera voltages. However, such reflections were not a concern, since their influence was eliminated in the calibration process.

Noise observed in the camera system was larger than expected from the noise equivalent power of the detector alone. Some noise may be expected from variations in background lighting, though this source did not seem to explain the magnitude of noise observed. Oscilloscope waveforms suggested electro-magnetic noise pickup. With more careful shielding and grounding practices, the optical detector should be capable of resolution better than the best available CCD detectors.

§3.5 *Suggested Modifications*

Lower noise and immunity to LED intensity variations are possible. As a first step, a better regulated current driver for the LED's would result in less variation among LED intensities. Better still, the current driver could be designed to adjust the current for each LED dynamically to achieve a constant light power at the detector, regardless of the age, efficiency or location of the emitter. The sum of all currents through the four electrodes of the detector could be used as a feedback signal to adjust current levels in the LED's during pulses. No normalization would then be required, and each LED set would conform to a single calibration lookup table.

A larger field of view would have been desirable. However, large diameter lenses with focal lengths shorter than the one selected (28mm) are impractical, and using a larger focal length (i.e. placing the lens further from the plane of the LED's) results in a signal loss proportional to the square of the distance from the light sources. Some signal loss can be tolerated if the noise level is reduced as well. Better shielding and more careful grounding practices could be implemented to reduce the major source of noise: electro-magnetic interference.

Linearity was not an issue in the present system, since a calibration lookup table was used. In 3-D, though, a dense three-dimensional lookup table could be impractical. In that case, good linearity would be a desirable feature, since the lookup table could then be more sparse, or perhaps eliminated altogether. Better linearity could be achieved by using a more linear lens (larger focal length), and by using only the center region of the detector. Both of these changes would reduce the field of view, which would require placing the camera at a greater distance from the sensed work volume. To reduce the power loss from placing the LED's at a greater distance from the camera, LED's with a smaller emission angle would be required.

§4. Computing System

The computing system is the third and final major subsystem of the experimental apparatus. Its construction did not require much design. Commercially available modules were combined in a straightforward manner to produce a simple, yet powerful multiprocessing system. The effectiveness of the multiprocessor depends heavily on the programmer's proper use of the system. At least an overview knowledge of the system architecture is essential for efficient programming. No operating system was used; details of communications among processors had to be specified explicitly by the programmer.

The multiprocessor consisted of 5 single-board computers (PV682, Pacific Microcomputers, Cardiff, CA) on a common high-speed VME backplane (Dawn VME Products, Santa Clara, CA). Each of the computer boards contained a 16.7MHz MC68020 microprocessor, a MC68881 floating point coprocessor, 1 Megabyte of RAM, a VME-bus interface, and an additional private interface connector wired directly to the microprocessor. Roughly, the computing power of each of the boards was equivalent to a DEC VAX 780.

Only interprocessor communication was performed over the bus. All other I/O was performed through custom electronics interfaced to the private connectors. Two such interface boards were constructed. An interface board with a multiplexed parallel interface was constructed for camera control. Signals from this board were used for turning currents on and off for each of 8 sets of LED's, for initiating conversions of detector voltages to digital form, for testing end-of-conversion status, and for reading the digitized camera data. The second interface board was used for robot I/O. It included two digital to analog converters used for sending analog commands to the PWM amplifiers, and two sets of logic circuitry for counting transitions from the robot's optical encoders. In addition, a 2MHz clock and counter were built on this board to obtain time information for control routines. The 16-bit counter continuously incremented with each clock pulse until its maximum value, then wrapped around to zero and started over. By sampling the counter at intervals no longer than the wraparound time (32.7ms), accurate time increments could be measured. This feature was essential in programs which made logical decisions resulting in actions of varying complexity. In such programs, no fixed time constant or sampling period could be pre-assigned. Thus, operations which required filtering, time differentiation or integration could only be performed by having access to a time base which was independent of the program execution.

Memory on each of the computer boards could be accessed off-board as well, via the VME bus. This feature was not used, however. Instead, a separate 8 Megabyte RAM board (CI-VMEMORY-8, Chrislin Industries, Canavanas, PR) was connected to the common backplane. This "global" memory was accessible by all boards for both reading and writing. All communications among processors were performed by reads and writes in global memory. Assigning and protecting memory space for communications was the user's responsibility.

The last component connected to the VME bus was a VME bus repeater (PT-VME-902, Performance Technologies, Inc, East Rochester, NY). The bus repeater connected the backplane of the multiprocessor with the backplane of a Sun 3/75 workstation (Sun Microsystems, Mountain View, CA). Bus arbitration was performed by the Sun and communicated to the multiprocessor bus via the repeater. The Sun workstation utilized an architecture nearly identical to that of the individual boards of the multiprocessor. The Sun 3/75 contained a MC68020 microprocessor, a MC68881 floating point coprocessor, 4 Mbytes of RAM, and a 2-slot VME backplane. Off-line storage was performed by a fileserver (Sun 3/260) over an Ethernet local area network.

The Sun workstation acted as the host and development system for the multiprocessor. Code written and debugged on the Sun was downloaded via the bus repeater into program memory of the multiprocessor boards. All code was written in "C" and compiled on the Sun's compiler. Since the processors of the Sun and Pacific Micro computers were identical, code compiled on the Sun ran on either machine. Thus, the operating system (Sun UNIX 4.2), program development tools and graphics of the Sun were available for writing and debugging software to run on the multiprocessor. The Sun was also useful for testing program operation in near real time. Actually, the Sun ran somewhat faster, since it used fast cache memory and a 20MHz processor. On the other hand, the multi-tasking UNIX operating system caused occasional interrupts of programs running on the Sun, making the Sun unsuitable for time-critical control programs. For programs in which brief (1ms) interruptions were not a problem, the Sun could act as a sixth computer in the multiprocessing system. All global memory was accessible by the Sun as well as by the Pacific Micro computer boards.

For the experiments described in this document, the individual computer boards were programmed to execute separate but coordinated tasks, as described in here in separate chapters. The first processor acted as a sensing computer, as described in this chapter, which controlled the timing, data acquisition and processing of position information from the electro-optical sensor. The second board performed robot servo control using the sliding-mode time-optimal control theory presented in chapter 3. The third board used the data from the sensory board, as deposited in global memory, to rapidly compute and update configuration space transforms for moving obstacles, as described in chapter 4. A map of obstacle boundaries in configuration space was maintained in global memory, accessible to other processors. The fourth processor executed reflex control, as described in chapter 5, which examined local regions of the configuration space map for possible danger, and prevented the robot from entering forbidden regions. The last processor executed local planning, as described in chapter 6. The local planner also accessed the configuration space map in global memory, and provided short-range paths for the robot to escape from local stall points (local energy minima) of the reflex controller. The Sun workstation was used for diagnostics and real-time graphics while the robot was running. A display of evolving configuration space, as read from global memory, was continuously updated on the Sun graphics monitor, along with the current position of the robot and its goal. Communications among the processors were also monitored and displayed by the Sun during robot control.

§5. Conclusion

In this chapter, the motivation and design details for the three subsystems of an experimental robot system have been presented. In each subsystem, simplifications were made either by restricting the problem domain or by overdesigning the components for their task. It was the intent to design a system which would expose fundamental issues and impediments in high-performance robot control.

A simple, but very fast and stiff 2 degree-of-freedom planar robot arm was designed and built using the principles of (virtual) direct-drive, remote drive, and dynamic decoupling. High accelerations, low friction, zero backlash and high resonant frequencies were designed in, since each of these effects are known to influence the controlled performance of electromechanical systems.

A non-contact, high-bandwidth optical sensing system was implemented to detect locations of moving obstacles and moving goals. Generality was sacrificed for performance. Complex and time-consuming image processing operations for object identification and location were sidestepped by marking each of the objects to be sensed with LED light beacons. Further, object motions were restricted to a plane at a known distance from the detector.

A powerful multiprocessing system was assembled for sensor and servo control. The computing system was intentionally overdesigned so as to avoid encountering control limitations due to processing delays. While efficiency of control software is a basic issue, processing limitations of common computing systems would have posed an artificial and rapidly changing constraint.

Although each of the subsystems was designed for idealized experiments, the technology employed in each is applicable to current manufacturing systems. The experimental arm is limited to motion in a plane, and has been designed to carry virtually no payload. The principles of the design, however, are applicable to more general machines. Further, even the two-degree-of-freedom machine has applications, e.g. in laser cutting, water-jet cutting, and perhaps small parts handling (e.g., electronic components). The optical sensing system is by no means as general as a machine vision system. However, it too has industrial applicability. Although it would be impractical to mark all components in an assembly line with light beacons, it is entirely feasible to instrument reusable conveyor palates and parts feeders. A simple analog optical system sensing light beacons can easily be cheaper, faster, and more reliable than a full vision system. The multiprocessor is the most immediately applicable subsystem of the three. It uses technology which, while state of the art, is commercially available at reasonable cost. A user-friendly multiprocessor operating system would make the system more useful to the casual programmer, but the current system is certainly immediately applicable in industrial control.

Using the hardware described here, empirical investigations and verifications of theoretical developments in robot control were performed. Elements of the experimental apparatus emulate anticipated high-performance systems of the future: fast robots, efficient vision systems and powerful computing systems. In avoiding the known technological limitations of robot control, more fundamental issues were exposed. Specifically, given fast sensing, high-speed robots and powerful computers, how can a robot be made to execute fast, safe, and effective motions? How should the control be organized? What principles of classical and modern control theory apply, and what types of new theory are needed? Given powerful elements, how can an integrated system realize the full potential of its combined components. In the following text, some partial answers to these questions are presented rigorously. Some broader, speculative generalizations are inferred from experimental results on the present hardware. Most of the area remains an open field of research, in which the current work represents a single data point.

CHAPTER III

ROBUST NEAR TIME-OPTIMAL CONTROL

§1. Introduction

In this chapter, theoretical developments and experimental evidence are presented for a new realization of nearly time-optimal control. The technique involves the combination of traditional "bang-bang" time optimal control with the methods of sliding-mode control. The result is a nonlinear feedback scheme for maintaining a desired functional relationship among dynamic variables, in this case imitating the idealized dynamics of time-optimal control. The approach is nearly time optimal rather than exactly time optimal, since the bang-bang control components are restricted to values below full actuator saturation, which allows some actuator overhead for servoing to a set of desired dynamics. In addition, the bang-bang regime of control is blended with linear control near a stable goal location. Lyapunov stability proofs are given for the proposed controller. Results of implementation on a planar, dynamically-decoupled, pseudo direct-drive robot are presented.

§2. Bang-Bang Control

The simplest interesting case of time-optimal control is exhibited in second order systems, e.g. a pure inertia driven by an actuator with effort saturation. Actuator saturation is a mathematically necessary requirement for a time-optimal solution to exist. In the time-optimal solution, the actuator is driven in full saturation at all times, a characteristic which motivated the name "bang-bang control", [22]. This is in sharp contrast to linear control, in which it must be assumed that actuators never saturate in order for the linear equations to remain valid.

Figure 1 illustrates the time-optimal control solution for a step command of a simple second order system described by:

$$\begin{aligned} I\ddot{\theta} &= u \\ \text{where} \quad |u| &\leq u_{max} \end{aligned} \tag{III.1}$$

In the example of figure 1, the time-optimal control solution is illustrated for such a system with initial conditions $\theta(0) = -1$ rad, $\omega(0) = 0$ and final conditions $\theta(t_f) = 0$, $\omega(t_f) = 0$. The optimal torque history consists of maximum acceleration towards the goal, switching to maximum braking midway between the initial and final positions. The velocity ramps up linearly until the switch-point, then ramps down again. The position increases quadratically until the midpoint and blends to a second quadratic function at an inflection point which occurs at torque switching. In this ideal scenario, the transition is made perfectly: the system reaches the goal in absolute minimum time without overshoot.

A phase-plane representation of optimal control for this system is shown in figure 2. A convenient coordinate frame is chosen in which the desired final state is described as the origin of the phase plane, $\theta(t_f) = 0$, $\omega(t_f) = 0$. Time-optimal control solutions for arbitrary initial conditions are then expressed in the phase plane in terms of a switching curve. For all states "above" the switching curve, maximum negative control effort is exerted. For all states below the curve, maximum positive control effort is exerted. The equation of the switching curve is defined implicitly by:

$$s(\theta, \omega) \equiv \theta + \frac{\omega|\omega|}{2u_{max}/I} = 0 \tag{III.2}$$

The phase-plane description of optimal control suggests a nonlinear state-variable

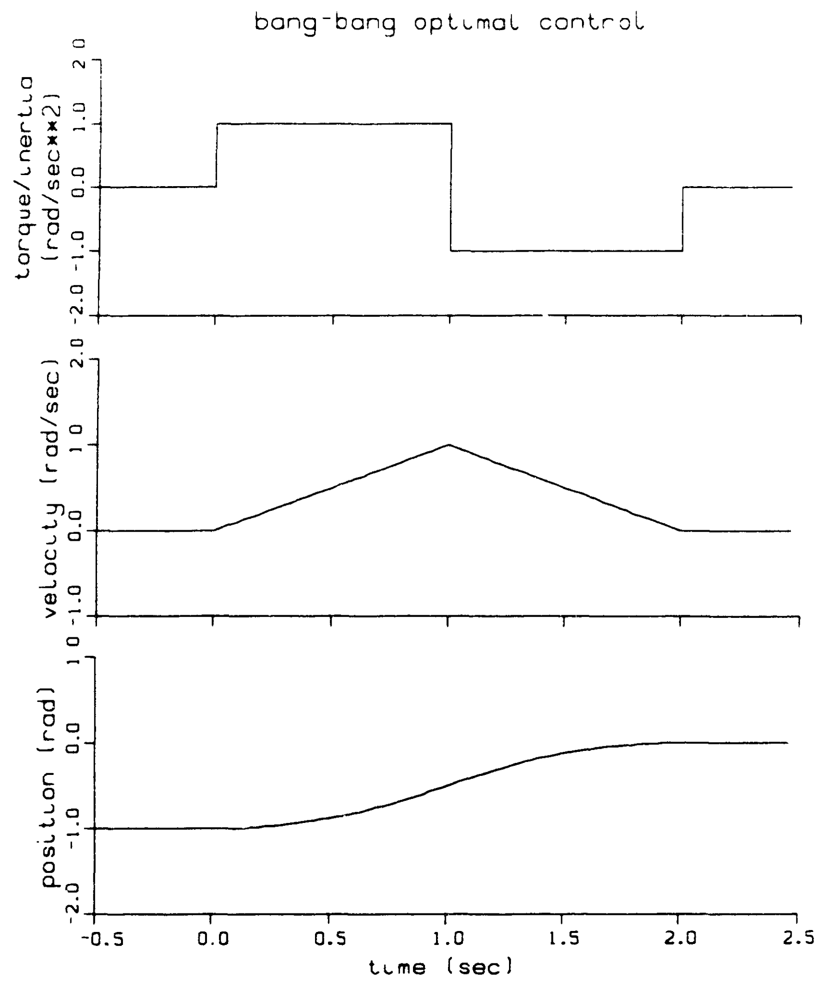

Figure 1: Bang-Bang Control of a Pure Inertia

feedback law [22]:

$$
\begin{array}{rll}
\text{above the switching curve:} & u = +u_{max} & \text{for } s > 0 \\
\text{below the switching curve:} & u = -u_{max} & \text{for } s < 0 \\
\text{on the switching curve:} & u = +u_{max} & \text{for } s = 0 \text{ and } \omega < 0 \\
 & u = -u_{max} & \text{for } s = 0 \text{ and } \omega > 0
\end{array}
\qquad \text{(III.3)}
$$

The state-variable feedback law enables optimal control implementation in a closed loop form, rather than a pre-computed ballistic open loop form.

Bang-bang control is useful for establishing a theoretical bound on the best possible controlled system performance. Implementation of optimal control, how-

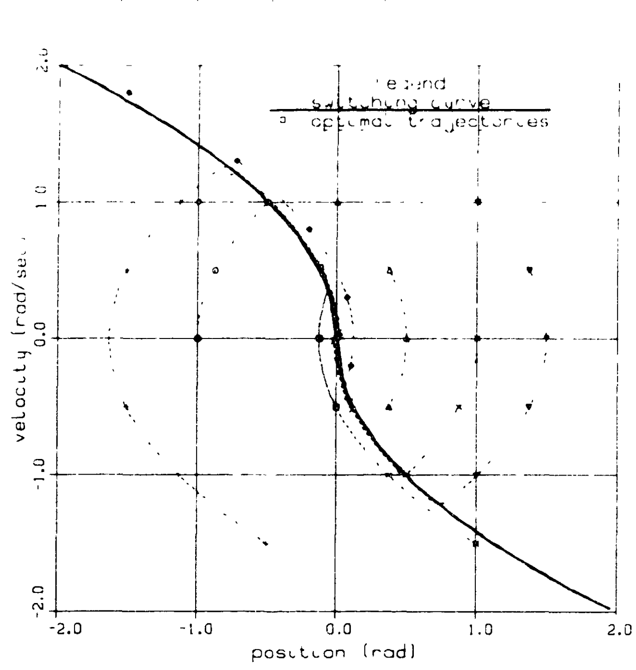

Figure 2: Phase Plane Optimal Switching Curve

ever, is generally impractical. The effectiveness of executing pre-computed torque histories suffers from modeling inaccuracies and unpredicted disturbances. Especially during long moves, minor errors in inertia estimates or friction estimates, minor external disturbances and minor timing errors can result in substantial trajectory errors [32]. Sensitivity to modeling errors is reduced by using a nonlinear state-dependent feedback law, (equation III.3), based on a pre-computed switching curve, (equation III.2). However, implementation of the nonlinear feedback law is typically impractical. Sensitivity to parameter estimation is less severe, but the presence of unmodeled dynamics, e.g. mechanical vibration modes, can easily make the feedback unstable. Even when stability is achieved, the results can be unsatisfying; the system tends to "chatter" near the switching curve, and parameter estimate errors can still lead to substantial overshoot. In addition, bang-bang control is not well behaved at the goal state. Chatter, characteristic near the switching curve, also occurs at the goal point.

The virtue of bang-bang control is its time optimality. Sensitivity to parameter uncertainty and unmodeled dynamics, and violent behavior along the switching curve and at the goal point are features which must be addressed to make bang-bang control practical. A technique for achieving these aims is presented next.

§3. Sliding Mode Optimal Control

Nonlinear state variable feedback for optimal control based on the equation of a switching curve suggests a sliding-mode, [6,46,47,53], implementation. In equation III.3, the function $s(\theta, \omega)$ describes a switching curve implicitly by the relation $s = 0$. The relation $s = 0$ also represents a description of the desired dynamics. That is, if one could maintain the relation $s = 0$, the system trajectory would correspond to motion along the switching curve. Such motion is exactly the perfect dynamics for completing the theoretically time-optimal trajectory. The objective of the sliding-mode implementation, then, is to force convergence to the relation $s = 0$ as quickly as possible, then maintain the relation $s = 0$ in a stable and robust manner.

The analysis here parallels that of Slotine in [6,46,47]. In the examples given by Slotine, the equation of desired dynamics is chosen to imitate a linear system. Nonlinear feedback is then applied to force the system to exhibit the chosen set of dynamics. In the present case, the desired dynamics are nonlinear, based on the time-optimal control solution, and sliding-mode feedback is applied to imitate the time-optimal dynamics.

In order to exert sliding-mode corrections, the nominal control history must stay within the actual saturation bounds. That is, some actuation overhead must be available to drive the system back to $s = 0$ when modeling errors or disturbances cause a deviation. Thus, a nearly time-optimal solution is computed based on an underestimate of the true saturation effort. The nominal dynamics to be achieved are expressed as:

$$\hat{s}(\theta,\omega) \equiv \theta + \frac{\omega|\omega|}{2\hat{u}/I} = 0 \tag{III.4}$$

where $\hat{u}$ is an underestimate of the actual saturation torque. It is used to express the bang-bang actuation levels corresponding to the theoretical minimum-time solution, given the restriction that the control effort may not exceed the chosen limit, $\hat{u}$.

The first proposed feedback law is:

$$u = -\hat{u}[\operatorname{sgn}(\omega) + \epsilon \operatorname{sgn}(\hat{s})] \tag{III.5}$$

To consider the effects of parametric modeling errors, the term $\tilde{I}$ is introduced. Evaluation of the switching function depends on a knowledge of the system inertia, I. An imperfect estimate of I results in an imperfect computation of s, denoted as $\tilde{s}$. It will be assumed that state measurements are performed accurately. The value of $\tilde{s}$ used in the computation of the control policy is:

$$\tilde{s}(\theta,\omega) \equiv \theta + \frac{\omega|\omega|}{2\hat{u}/\tilde{I}} = 0 \tag{III.6}$$

The first term in the computation of control effort mimics the control policy for bang-bang optimal control along the switching curve $s = 0$. Since an underestimate of the saturation effort has been employed, an additional term is included to utilize the overhead of remaining actuator effort to achieve and regulate the condition $\tilde{s} = 0$.

Uncertainty in the estimated inertia has the effect of making the chosen set of dynamics deviate from the theoretically perfect set of dynamics. Enforcing the condition $\tilde{s} = 0$ forces the system to imitate a pure inertia of magnitude $\tilde{I}$. Under open-loop bang-bang control, such an error would result in over or undershoot of the goal. With the reserved effort overhead, however, corrections can be made to make the system imitate the erroneous inertia. An incorrect estimate of inertia does not affect stability; rather, it affects the choice of desired dynamics. If that choice deviates significantly from reality, then a correspondingly large torque overhead must be allocated to compensate.

The chosen dynamics $s = 0$ can be interpreted intuitively in terms of a braking distance. If a pure inertia of magnitude I is moving with a velocity ω, then an actuation effort of u_{max} can bring the system to rest in a braking distance of $\omega^2 I/u_{max}$. Thus, if $s = 0$, then a braking control effort of u_{max} should bring the system to rest as it coasts to the origin. Such behavior corresponds to following the optimal control switching curve to the origin. Following the imperfect law $\tilde{s} = 0$ corresponds to tracking an estimate of the switching curve.

The proposed control scheme can be proven to converge on the dynamics described by $\tilde{s} = 0$. The proof follows. The function defined by $V_1 = |\tilde{s}|$ is proposed as a Lyapunov function. It is positive semidefinite (in $\tilde{s}$) and is equal to zero only at the desired final condition, $\tilde{s} = 0$. To determined whether V_1 satisfies the properties of a Lyapunov function, examine $\dot{V}_1$.

$$\dot{V}_1 = \dot{\tilde{s}}\,\mathrm{sgn}(\tilde{s}) \tag{III.7}$$

Evaluate $\dot{s}$ as follows:

$$\begin{aligned} \dot{\tilde{s}} &= \frac{1}{2\hat{u}/I}[\dot{\omega}|\omega| + \omega\,\mathrm{sgn}(\omega)\dot{\omega}] + \omega \\ &= \frac{u/I}{\hat{u}/I}|\omega| + \omega \end{aligned} \tag{III.8}$$

Substituting in the proposed control law $u(\tilde{s}, \omega)$ yields:

$$\dot{\tilde{s}} = \omega[1 - \tilde{I}/I] - \epsilon|\omega|\mathrm{sgn}(\tilde{s})(\tilde{I}/I) \tag{III.9}$$

Substituting $\dot{\tilde{s}}$ from equation III.9 into equation III.7 for $\dot{V}_1$ yields:

$$\dot{V}_1 = \mathrm{sgn}(\tilde{s})\omega(1 - \tilde{I}/I) - \epsilon|\omega|(\tilde{I}/I) \tag{III.10}$$

In order to prove global stability, it is sufficient to guarantee that $\dot{V}_1 \leq 0$, where equality holds only at the desired stable point. This relation can be guaranteed by making epsilon large enough. In specific:

$$\epsilon > \left| \frac{I}{\tilde{I}} - 1 \right| \qquad \text{(III.11)}$$

Under the preceding condition, the system will converge on $\tilde{s} = 0$, and imitate optimal control of a system with inertia $\tilde{I}$ and saturation control effort $\hat{u}$. Implicit in the proof is that the actuators are capable of exerting the control efforts determined by the proposed control law. This restriction will be relaxed later. First, though, the proposed control law will be modified to make it smoother and more robust.

§4. Sliding Mode Control with Smoothing

The proposed nonlinear control law has been shown globally stable and insensitive to parametric errors. However, it has not been shown to be insensitive to unmodeled dynamics, nor is the resulting behavior guaranteed to be chatter-free. An improved control law is:

$$u = -\hat{u}[\mathrm{sat}(\omega/\omega_{sat}) + \epsilon\,\mathrm{sat}(s/s_{sat})] \qquad \text{(III.12)}$$

The preceding control law invokes the saturation function, "sat" which is defined as:

$$\begin{array}{ll} \text{for all } |x| > 1: & \mathrm{sat}(x) = \mathrm{sgn}(x) \\ \text{for all } |x| \leq 1: & \mathrm{sat}(x) = x \end{array} \qquad \text{(III.13)}$$

In the region $\omega > \omega_{sat}$ and $\tilde{s} > s_{sat}$, the new control law behaves the same as the previous control law: $\tilde{s}$ is driven monotonically towards zero. As $\tilde{s}$ or ω decline below saturation, the behavior differs from that of the original sharply-switching controller. Three cases need to be considered: 1) $\omega < \omega_{sat}$ and $\tilde{s} > s_{sat}$; 2) $\omega > \omega_{sat}$ and $\tilde{s} < s_{sat}$; and 3) $\omega < \omega_{sat}$ and $\tilde{s} < s_{sat}$. It will be convenient to redefine the functional saturation values of ω and $\tilde{s}$ as:

$$\begin{aligned} \omega_{sat} &= \hat{u}/b \\ s_{sat} &= \epsilon\hat{u}/K \end{aligned} \qquad \text{(III.14)}$$

For Case 1, the control law of equation III.12 may be written as: $u = -\hat{u}[\omega/\omega_{sat} + \epsilon\,\text{sgn}(\tilde{s})]$. Substituting this expression for u into equation III.8 yields a modified version of equation III.9:

$$\dot{\tilde{s}} = \omega[1 - (|\omega|/\omega_{sat})(\tilde{I}/I)] - \epsilon|\omega|\text{sgn}(\tilde{s})(\tilde{I}/I) \qquad \text{(III.15)}$$

Invoking equation III.15 in equation III.7 results in:

$$\dot{V}_1 = \text{sgn}(\tilde{s})\omega[1 - (|\omega|/\omega_{sat})(\tilde{I}/I)] - \epsilon|\omega|(\tilde{I}/I) \qquad \text{(III.16)}$$

Convergence to $\tilde{s} = 0$ is guaranteed if $\dot{V}_1 < 0$, which is true according to equation III.16, provided $|\omega| < \omega_{sat}$, (which is true for Case 1), and provided ϵ is chosen such that:

$$\epsilon > I/\tilde{I} \qquad \text{(III.17)}$$

Given the above, Case 1 conditions result in a convergence of $\tilde{s}$ to 0. Thus, Case 1 eventually evolves into Case 2 or Case 3.

Behavior of the system under the conditions 2) and 3) will be analyzed with respect to a new candidate Lyapunov function, V_2:

$$V_2 = \tilde{I}\frac{\omega^2}{2} + K\frac{\theta^2}{2} \qquad \text{(III.18)}$$

The proposed Lyapunov function is positive semi-definite, with $V_2 = 0$ only at the origin, $\omega = 0$, $\theta = 0$. The time derivative, $\dot{V}_2$, is:

$$\begin{aligned} \dot{V}_2 &= \tilde{I}\omega\dot{\omega} + K\theta\dot{\theta} \\ &= \omega u + K\theta\omega \end{aligned} \qquad \text{(III.19)}$$

Equation III.19 may be evaluated by substituting for u from equation III.12. For Case 2, the control law may be written as: $u = -\hat{u}[\text{sgn}(\omega) + \epsilon\tilde{s}/s_{sat}]$. In this regime, substituting for u in equation III.19 yields:

$$\begin{aligned}\dot{V}_2 &= -\omega\hat{u}\,\mathrm{sgn}(\omega) - \omega\hat{u}\epsilon(\tilde{s}/s_{sat}) + K\theta\omega \\ &= -|\omega|\hat{u} - K\tilde{s}\omega + K\theta\omega \\ &= -|\omega|\hat{u} - K\omega(\theta + \tilde{I}\omega|\omega|/(2\hat{u})) + K\theta\omega \\ &= -|\omega|\hat{u} - K\omega^2|\omega|\tilde{I}/(2\hat{u})\end{aligned} \tag{III.20}$$

Equation III.20 proves that, under the conditions of Case 2, $\dot{V}_2$ satisfies sufficient conditions for global stability of the proposed controller: $\dot{V}_2 \leq 0$ with no trajectories in $\dot{V}_2 = 0$.

For the conditions of Case 3, the control law corresponds to: $u = -\hat{u}[\omega/\omega_{sat} + \epsilon\tilde{s}/s_{sat}] = -b\omega - K\omega|\omega|\tilde{I}/(2\hat{u}) - K\theta$. In this regime, $\dot{V}_2$ is:

$$\begin{aligned}\dot{V}_2 &= \omega u + K\theta\omega \\ &= -b\omega^2 - K\omega^2|\omega|\tilde{I}/(2\hat{u})\end{aligned} \tag{III.21}$$

Case 3 results in V_2 decreasing faster than a polynomial in ω^2, and no trajectories lie in $\dot{V}_2 = 0$ except the origin.

The preceding proofs show that Case 1 conditions will force $\tilde{s}$ towards 0, during which the system will enter Case 2 or Case 3. Once in Case 2 or Case 3, the system will converge asymptotically to the origin.

§5. Analysis of Overshoot

Overshoot of a goal location is often an important consideration, e.g. when the goal position abuts a hard or fragile surface. Ideally, zero overshoot would occur, but at the least any overshoot should be predictable so that a safe subgoal may be specified. For linear systems, critical damping is the typical approach to preventing overshoot. For the present nonlinear system, analysis of linear control can be invoked for a bounding behavior of the nonlinear controller.

In the proposed smoothed controller, equation III.12, the values of s_{sat} and ω_{sat} were defined in terms of K and b in order to interpret the behavior of the system under the conditions of Case 3: $\omega < \omega_{sat}$ and $\tilde{s} < s_{sat}$. The control law in this case resembles a proportional plus derivative linear feedback,($u \approx -b\omega - K\theta$), plus an additional non-linear term proportional to ω^2. The values of K and b may be selected based on linear control principles; the third, nonlinear term provides additional damping. Thus, K and b may be selected to achieve critical damping based on linear feedback; the additional nonlinear term can not increase overshoot beyond that predicted by linear control.

For short moves, the nonlinear controller of equation III.12 will behave like a linear controller, so the selection of b for critical damping will prevent overshoot. For a long move, however, the nonlinear controller behaves differently. The condition $\tilde{s} = 0$ will be regulated, and the goal will be approached with a nominal braking control effort of magnitude $\hat{u}$. As the approach velocity falls to within ω_{sat}, nearly-linear control will be exerted. Assuming the nonlinear controller successfully regulates $\tilde{s} \approx 0$, the initial conditions upon entering the nearly-linear regime will satisfy:

$$\theta_0 + \omega_0|\omega_0|\tilde{I}/(2\hat{u}) = 0 \qquad \text{(III.22)}$$

and

$$|\omega_0| = \omega_{sat} \qquad \text{(III.23)}$$

Employing the definition of ω_{sat} and using the definitions $\omega_n = \sqrt{K/\tilde{I}}$ and $b/\tilde{I} = 2\zeta\omega_n$, where $\zeta = 1$ for critical damping, results in the initial conditions in terms of control parameters:

$$\theta_0 = \text{sgn}(\theta_0)\frac{\hat{u}/\tilde{I}}{8\omega_n{}^2} \qquad \text{(III.24)}$$

and

$$\omega_0 = -\text{sgn}(\theta_0)\frac{\hat{u}/\tilde{I}}{2\omega_n} \qquad \text{(III.25)}$$

Upon entering the regime of Case 3, the initial conditions satisfy:

$$\omega_0 = -4\omega_n\theta_0 \qquad \text{(III.26)}$$

Although the control parameters K and b may have been picked for critical damping, zero overshoot is only guaranteed if the initial approach velocity is below a limit based on ω_n. To find the limiting zero-overshoot approach velocity, consider the solution of a critically damped linear system:

$$\theta(t) = \theta_0 e^{-\omega_n t} + t(\omega_0 + \theta_0 \omega_n) e^{-\omega_n t} \tag{III.27}$$

where θ_0 and ω_0 are the initial conditions. For zero overshoot, the second term on the right side of equation III.27 must have the same sign as the first term. The maximum approach velocity for zero overshoot is thus $\omega_0 = -\theta_0 \omega_n$. Equation III.26 indicates that the approach velocity at $\omega = \omega_{sat}$, $\tilde{s} = 0$ is four times larger than the limit for zero overshoot with critical damping.

For the initial conditions of equations III.24 and III.25, linear control with critical damping would result in an overshoot of $(0.770)\theta_0$. A solution of the nonlinear controller with $\varsigma = 1$ was obtained numerically for the initial conditions of equations III.24 and III.25. With the nonlinear damping term, overshoot is somewhat lower, at $(0.6825)\theta_0$. For ω_n chosen large, the value of θ_0, the position at which bang-bang control blends with linear control, will be small, and the subsequent overshoot will be correspondingly small.

§6. Influence of Unmodeled High-Frequency Dynamics

It has been shown that the control law of equation III.12 is asymptotically stable in all regimes. Further, the definitions of s_{sat} and ω_{sat} allow an interpretation of the choice of b and K in terms of linear control. The result is a blend of (nearly) bang-bang control and (nearly) linear control.

The value of b in the definition of ω_{sat} may be selected to choose damping based on the bounding linear system behavior. The value of K also has a linear interpretation. By increasing K, the time constant of convergence in the (nearly) linear region is decreased. This is equivalent to increasing the proportional gain.

For an ideal second-order linear system, the proportional gain may be increased without bound and stability will be preserved. In reality, high-frequency dynamics, which were not included in the second order system model, will cause the system to go unstable at high gain. Thus, the choice of K for stable control is limited by unmodeled dynamics.

As an illustration of the influence of unmodeled dynamics, consider the idealized system of equation III.1, which has a transfer function given by:

$$\frac{\theta}{u} = \frac{1}{Js^2} \tag{III.28}$$

For a linear control law given by:

$$u = -b\omega - K\theta \tag{III.29}$$

the characteristic equation of the closed loop system is:

$$s^2 + (b/J)s + (K/J) = 0 \tag{III.30}$$

The feedback parameters define a closed-loop frequency, $\omega_f = \sqrt{K/J}$, and closed-loop damping, $2\varsigma_f\omega_f = b/J$. For all positive values of b and K, the roots of the characteristic equation have positive real parts, which implies that the closed-loop system will be stable. As long as the model remains valid, the feedback coefficients may be increased without bound and stability will be preserved. At high bandwidths, however, the system will exhibit high frequency dynamics which were not included in the system model. To examine the effects of unmodeled dynamics, consider the augmented transfer function:

$$\frac{\theta}{u} = \frac{1}{Js^2}\frac{{\omega_u}^2}{s^2 + 2\varsigma_u\omega_u + {\omega_u}^2} \tag{III.31}$$

In the above, the second factor on the right is a term which introduces two more dynamic variables into the transfer function. At frequencies well below the natural frequency of the unmodeled dynamics, ω_u, the transfer function of equation III.31 closely matches that of equation III.28. Only at high frequencies are the additional dynamics apparent. Now, using the feedback law of equation III.29 results in a characteristic equation of:

$$s^4 + 2\varsigma_u \omega_u s^3 + s^2 {\omega_u}^2 + s 2\varsigma_f \omega_f {\omega_u}^2 + {\omega_f}^2 {\omega_u}^2 = 0 \qquad \text{(III.32)}$$

The roots of the characteristic equation determine the poles of the closed-loop system. In order for the system to be stable, all of the poles must have negative real parts. For the transfer function of equation III.30, stability only requires $K > 0$ and $b > 0$. For the transfer function of equation III.32, two additional conditions must be satisfied:

$$\omega_f < \omega_u \frac{\varsigma_u}{\varsigma_f} \qquad \text{(III.33)}$$

and

$$\omega_f < \omega_u \frac{\varsigma_u \varsigma_f}{{\varsigma_u}^2 + {\varsigma_f}^2} \qquad \text{(III.34)}$$

The largest allowed value of ω_f occurs for $\varsigma_f = \varsigma_u$, (which may not be a desirable choice of ς_f in terms of system overshoot), at which the maximum stable value of ω_f is one half ω_u. For other choices of ς_f (e.g., critical damping), the maximum stable closed-loop natural frequency, ω_f, may be substantially lower than one half ω_u. For any non-zero choice of b, (or ς_f), it it is always possible to choose a K, (i.e., a ω_f), low enough to satisfy the stability criteria. The high-frequency dynamics introduced in equation III.31, however, impose an upper bound on the choice of K.

In section 3, it was proven that the controller of equation III.5 is globally stable. In section 4, smoothing was added by changing signum functions to saturation functions. It can be shown that this change also increases robustness with respect to unmodeled dynamics. Near the goal state, the choice of K, (i.e., the choice of s_{sat}), determines the effective proportional feedback gain. As K is increased to infinity, (or as $s_{sat} \to 0$), the saturation function of $\tilde{s}$ approaches the behavior of a signum function. Thus, near the origin, signum functions behave like extremely high gain proportional feedback. As illustrated in the preceding analysis, though, the maximum permissible proportional feedback gain is limited by unmodeled high-frequency dynamics. The former proof of stability for the controller of equation III.5, while mathematically rigorous, is deceptive. It depends on the assumption that the system equation includes all dynamics. Any system model, however, is only a truncated approximation of the real dynamics.

With the smoothing functions of controller equation III.12, robustness to unmodeled dynamics is incorporated. The choice of K determines the equivalent linear proportional feedback. The equivalent linear natural frequency of the closed-loop system is $\omega_f = \sqrt{K/I}$. For choices of ω_f well below the first natural frequency of unmodeled dynamics, ω_u, the system model of equation III.1 will be a good approximation of the actual system over the bandwidth of the controller. Under these conditions, the stability proof based on the assumed model with truncated dynamics, although not fully rigorous, will give the correct stability result. As illustrated above, the value of K may be chosen low enough (at sacrifice of system performance) to select a ω_f for which high-frequency dynamics do not destabilize the controller.

§7. Sliding Mode Control Saturation

It has been shown that the controller of equation III.12 is stable and robust. Within the linear domains of the saturation functions, the prescribed control effort is guaranteed to be feasible, since the control effort does not exceed $\hat{u}$, a parameter intentionally chosen to be less than the true actuator saturation level, u_{max}. In the stability proof of section 5, however, it was implicitly assumed that the control effort of equation III.12 could be realized. This condition can be met by choosing ϵ and $\hat{u}$ such that:

$$u_{max} > \hat{u}(1+\epsilon) \qquad \text{(III.35)}$$

Given the above, the computed control effort is always realizable. However, such a restriction on ϵ and $\hat{u}$ is undesirable. For the system response to approach that of true time-optimal control, $\hat{u}$ should be close to u_{max}. At the same time, to have good regulation of the condition $s = 0$, ϵ should be set as large as stability permits. Under these conditions, the exerted control effort will saturate at u_{max} although the control law may exceed this value. If the prescribed control law is not enforced, the stability proof is invalidated. With saturation, the control law of equation III.12 will behave according to:

$$u = -u_{max}\,\text{sat}\{\hat{u}/u_{max}[\text{sat}(\omega/\omega_{sat}) + \epsilon\,\text{sat}(\tilde{s}/s_{sat})]\} \tag{III.36}$$

It will be shown that this physically saturated control law is also stable. The previous proofs apply when the argument of the outermost saturation function is less than 1, i.e. when the result of equation III.36 is less than u_{max}. Proof extensions are required for cases where the outermost saturation function is in its nonlinear region, in which case:

$$1 < |\text{sat}(\omega/\omega_{sat}) + \epsilon\,\text{sat}(\tilde{s}/s_{sat})| \tag{III.37}$$

For the above inequality to be true, the expression $\text{sat}(\omega/\omega_{sat}) + \epsilon\,\text{sat}(\tilde{s}/s_{sat})$ must have the same sign as $\tilde{s}$. Since the first term saturates at 1, ω saturation can never lead to control saturation by itself. If the second term on the right of equation III.37, (the term in $\tilde{s}$), has the same sign as ω, then control effort saturation can occur. Alternatively, if ϵ is greater than 1, then the second term can dominate the first term to result in control saturation. In either event, the expression $\text{sat}(\omega/\omega_{sat}) + \epsilon\,\text{sat}(\tilde{s}/s_{sat})$ is guaranteed to have the same sign as $\tilde{s}$ whenever the magnitude of the expression exceeds unity.

Invoking the preceding deduction, the control law of equation III.36 behaves like equation III.12 when not in saturation, and when in saturation it reduces to:

$$u = -u_{max}\,\text{sgn}(\tilde{s}) \tag{III.38}$$

The test function $V_1 = |\tilde{s}|$ has been shown to satisfy the Lyapunov criterion which guarantees convergence to $\tilde{s} = 0$, provided u is not saturated. The test can be extended to consider saturation. Equations III.7 and III.8 remain valid regardless of control saturation. Under control saturation, equation III.38 applies for the value of u in equation III.8. Substituting u from equation III.31 into equation III.8 results in:

$$\dot{\tilde{s}} = \frac{\tilde{I}/I}{\hat{u}}|\omega|(-u_{max}\,\mathrm{sgn}(\tilde{s})) + \omega \qquad \text{(III.39)}$$

Substituting equation III.39 into equation III.7 for $\dot{V}_1$ results in:

$$\dot{V}_1 = -(u_{max}/\hat{u})(\tilde{I}/I)|\omega| + \omega \qquad \text{(III.40)}$$

The first term on the right side of equation III.40 dominates the second term, provided:

$$\hat{u} < u_{max}\tilde{I}/I \qquad \text{(III.41)}$$

By choosing $\hat{u}$ to satisfy equation III.41, equation III.40 indicates that $\dot{V}_1 < 0$ as long as the control is saturated. This extends the stability proof for the proposed controller from physical saturation levels through the nearly linear regime about the origin.

Since stability in saturation is guaranteed, large choices of ϵ are permitted, even when $\hat{u}$ is chosen close to u_{max}. A large value of ϵ can force a rapid convergence to the condition $\tilde{s} = 0$, and maintain a tight regulation of the equality. In doing so, the controller will closely mimic true time-optimal control. When ϵ is large, the control law of equation III.36 behaves like the optimal control law of equation III.3. For the example of figure 1, the error in $\tilde{s}$ will be initially large, and, with large ϵ, the term in $\tilde{s}$ will dominate the term in ω. Control will begin with maximum effort, u_{max}, just as in the theoretically perfect case. As $\tilde{s}$ approaches zero, the system converges to a switching curve based on a nominal maximum control effort of $\hat{u}$. For $\hat{u} = u_{max}$, and for $\tilde{I} = I$, the system converges to the theoretically perfect switching curve. Near the nominal switching curve, (the switching curve based on a control limit of $\hat{u}$), braking of magnitude $\hat{u}$ will be exerted, due to the term in ω. The term is $\tilde{s}$ will act to servo the system to the equation $\tilde{s} = 0$. As the velocity decreases near the goal point, the control behavior will blend into linear, critically damped motion.

For the limit of $\hat{u} \to u_{max}$, $\epsilon \to \infty$, $\omega_{sat} \to 0$ and $s_{sat} \to 0$, equation III.36 becomes identical to the theoretically perfect control law of equation III.3. The ideal controller is intolerant of model uncertainty, however. An underestimate of u_{max} is used for $\hat{u}$ in the siding-mode controller in order to tolerate uncertainty in I. Non-zero values of ω_{sat} and s_{sat} are used to reduce chatter and to make the control stable for real systems which contain unmodeled dynamics. The value of ϵ may be chosen arbitrarily large, according to the preceding proof. However, unmodeled dynamics will also impose a practical upper bound on ϵ.

If a system existed which corresponded to a perfect second order model with perfectly known parameters, the sliding mode control law proposed here would degenerate to the theoretically perfect time-optimal control policy. For real systems, the proposed controller is robust and is nearly time optimal. Experimental results from implementation of equation III.36 on a mechanical system are presented next.

§8. Implementation Results

The proposed control policy, equation III.36, was implemented on a mechanical system: the two degree-of-freedom planar, dynamically decoupled, pseudo direct-drive robot described in Chapter 2. Since the robot was constructed such that the two links are dynamically independent (when expressed in absolute joint coordinates), the control of link 1 is independent of the control of link 2. Control results for link 1 are presented here.

Link 1 is driven through a pulley and steel band transmission with a reduction of 4:1. By pretensioning the steel band, backlash was eliminated while preserving low friction. Relevant system parameters for link 1 are summarized here, expressed with respect to the output of the transmission (motion of link 1). Measured friction was 0.30 N-m. The first resonant mode occurred at 212 Hz. The motor for link 1 was driven by a PWM amplifier operated in transconductance mode with current saturation. The current saturation corresponded to an output torque saturation of 13 N-m. Motor current (within saturation bounds) was measured to track commanded current over a bandwidth of 1.5kHz. Inertia with respect to the transmission output (i.e., link1 inertia + 16 times motor rotor inertia) was measured via dynamic tests to be $I = 0.047$Kg-m^2. To an unusually good approximation, the dynamics of the experimental system agreed with the model of equation III.1.

System control was performed on a single-board computer based on the Motorola 68020 running at 16.7 MHz. Position measurements were obtained from optical encoders on the motors. Velocity measurements were not directly available. Instead, velocity was inferred from an observer running on the control computer [see e.g. 27 and Appendix]. Observer dynamics introduce additional dynamics in the controlled system, which, like mechanical resonances, imposes limitations on the validity of the second order plant model.

A sliding-mode controller was implemented on the control computer, as per equation III.36. The value of $\hat{u}$ was selected to be 10.5 N-m, 20% below the full torque saturation of $u_{max} = 13$ N-m. The sliding-mode gain, ϵ, was set to 4. The values of ω_{sat} and s_{sat} were 1.8 rad/sec and 0.23 radians, respectively, which correspond to a linear-region proportional gain of $K = 185$ N-m/rad and damping of $b = 5.9$ N-m/(rad/sec). These values correspond in turn to a linear equivalent natural frequency of $\omega_n = 10Hz * 2\pi$ and critical damping ($\zeta = 1$). With these values of K, ω_n, b, $\hat{u}$ and $\tilde{I}$, the predicted overshoot ($0.6825\,\theta_0$) is 0.0048 radians.

Transient measurements were taken for a step command of 3 rad. An initial condition of $\omega = 0$ and $\theta = -3$ rad was imposed at the onset of control. The control computer measured the link positions, computed an observed velocity, computed torque commands according to equation III.36, sampled a real-time clock, and stored samples of position, velocity, torque and time in memory for

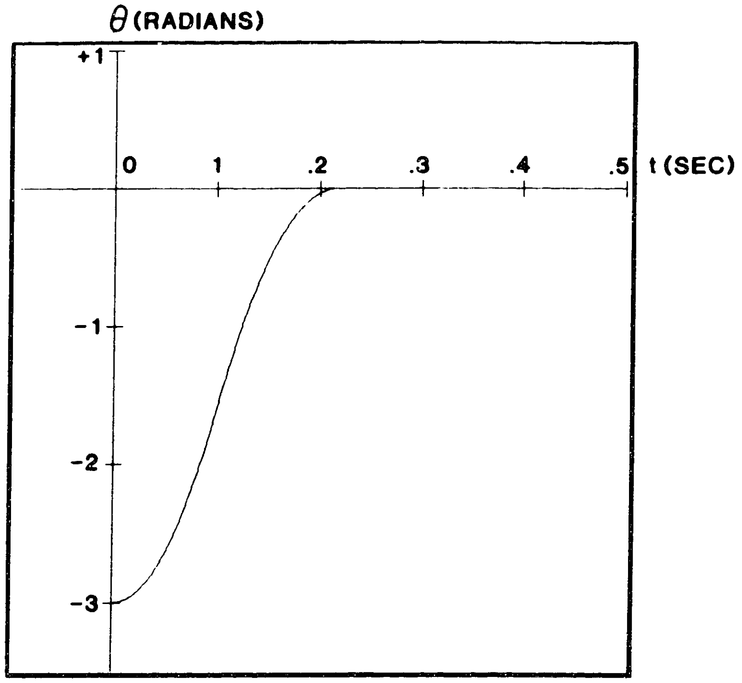

Figure 3: Robust Optimal Control; Angle vs Time

analysis. The cycle time of the routine was 660 microseconds. Position, velocity and torque histories are shown plotted in figures 3, 4 and 5. The time history of $\tilde{s}$ is plotted in figure 6. These results are physical measurements, not simulation.

The trajectory shown in figure 3 is similar to that of the ideal case shown in figure 1. The curve consists of a pair of quadratics which match the initial and final conditions, and which blend together at an inflection point near the middle of the transient. As expected, a slight overshoot did occur, although it is not apparent from the scale of the graph. The link first passed through the goal at time = 0.214 seconds. The overshoot peaked at 0.0017 radians, at time = 0.2195 seconds. Overshoot is lower than the predicted value of 0.0048; the difference is attributable to extra damping due to friction which was not included in the system model. During its return to the goal position, the link behaves like a 10Hz critically damped second-order system. The goal position was reached within 2 encoder counts (.0004 rad), by time 0.230 seconds. Very near the goal, non-idealities of coulomb friction and position measurement resolution make the

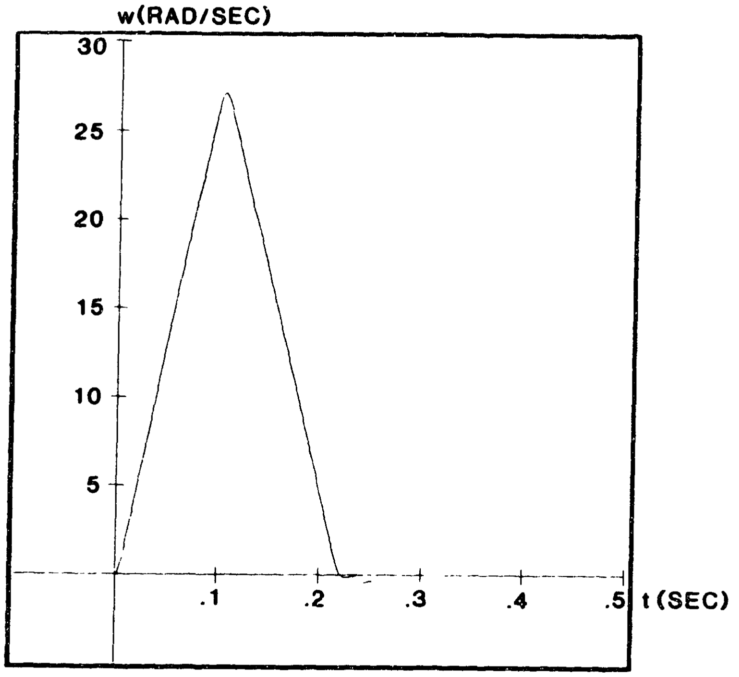

Figure 4: Robust Optimal Control; Velocity vs Time

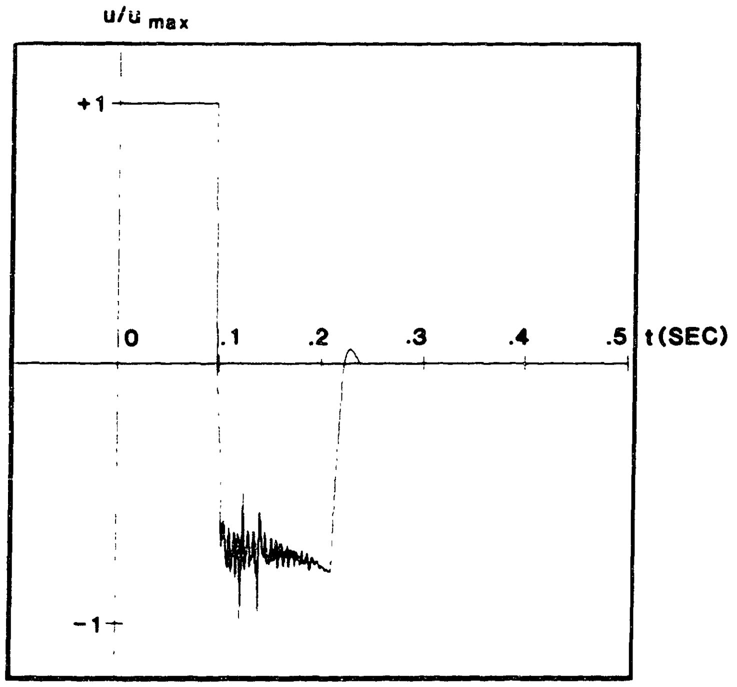

Figure 5: Robust Optimal Control; Normalized Torque vs Time

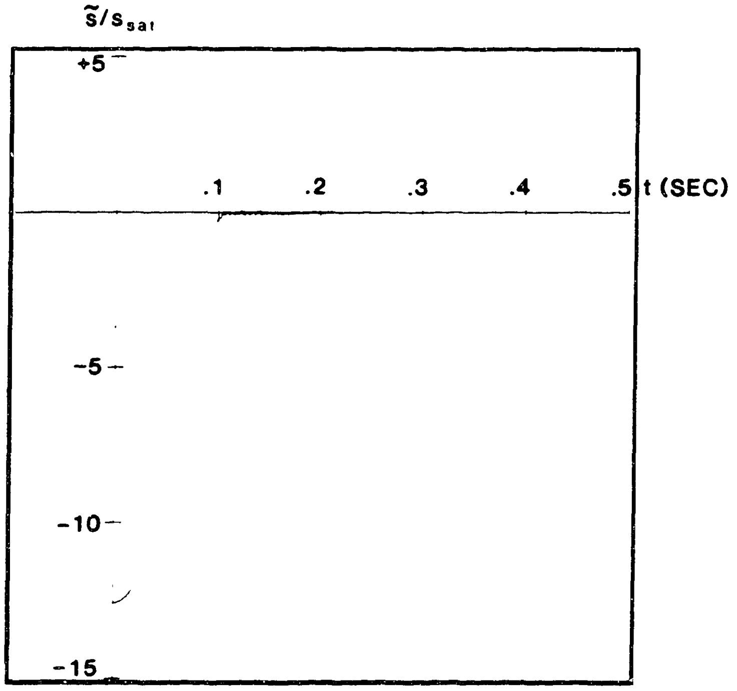

Figure 6: Robust Optimal Control; $\tilde{s}$ vs Time

system dynamics deviate from ideal linear behavior.

Deviations from ideal time-optimal control are more apparent in figures 4 and 5. Figure 4 shows a velocity profile similar to the ideal profile of figure 1, but with several distinguishing features. First, the slope of the velocity rise is steeper than that of the fall. The reason for this is apparent from the torque history of figure 5. Initially, the system dynamics are far from the desired relation $\tilde{s} = 0$; i.e. the initial state does not lie near the ideal switching curve. Maximum torque, u_{max}, is exerted to achieve the condition $\tilde{s} = 0$ as rapidly as possible. On the decelerating half, though, the condition $\tilde{s} = 0$ is achieved and regulated, so the braking torque exerted is close to $\hat{u}$, which has been chosen 20% below the true saturation torque. Thus, the link decelerates more slowly than it accelerates. The resulting behavior closely imitates that which would occur under ideal optimal control with a positive saturation torque of 13 N-m and a negative saturation torque of 10.5 N-m.

Another distinguishing feature of figure 4 is the smoothed transition from positive acceleration to negative acceleration. Ideal bang-bang control results in a sharp corner at the velocity peak. Smoothing introduced by the saturation functions in the sliding-mode control law blunts the velocity peak.

A third important feature of figure 4 is the velocity overshoot near the end state. Since the system overshot its goal position by 0.0017 rad, a negative velocity transient was produced by the linear-regime controller to return the system to the goal. The peak velocity in figure 4 is 27 rad/sec, which occurs at time = 0.098 seconds. The system first returned to zero velocity upon peak position overshoot at time 0.2195. The peak velocity resulting from the linear-regime controlled return to the goal point was -0.14 rad/sec.

The torque profile in figure 5 shows full saturated torque of 13 N-m exerted from onset to time 0.0985 sec, at which time the torque rapidly reverses. Under ideal time-optimal control, torque reversal would occur at time 0.104 seconds. The sliding-mode controller anticipates braking the system at -10.5 N-m instead of at full saturation; consequently, braking is initiated sooner than under time-optimal control. Also, under perfect bang-bang control the transition from maximum positive torque to maximum negative torque would occur instantaneously. With smoothed sliding-mode control, the transition occurred over 10 control loop iterations (about 7 ms).

Upon switching to braking torque, the sliding mode controller commands torques which vary about the nominal safe braking torque of -10.5 N-m. If the system model and all measurements were perfect, the braking torque exerted would be flat at -10.5 N-m. Due to nonidealities such as observer dynamics, parameter estimation errors, unmodeled friction, unmodeled resonances, finite computing speed and finite position measurement resolution, torque corrections about the nominal braking torque are required to maintain the condition $\tilde{s} = 0$. It is suspected that the dominant source of the chatter is due to the influence of discretized position measurements of the observer dynamics (see Appendix), although this has not been confirmed in the present study.

From figure 6, it is apparent that the controller is quite successful in enforcing $\tilde{s} = 0$. Initially, the systcm starts far from the switching curve. The initial value of $\tilde{s}/s_{sat}$ is -12.57. At the onset of torque reversal, the value of $\tilde{s}/s_{sat}$ has fallen to -0.662. By the time the rapid torque reversal is complete, (7ms later), the value of $\tilde{s}$ has fallen to 3% of saturation. Regulation of $\tilde{s} = 0$ continues to improve to better than 1% of s_{sat} over the duration of the move.

At time 0.212 sec, the link velocity falls below ω_{sat}, and the controller enters the near-linear regime in which both $\tilde{s}$ and ω are below saturation. The braking torque begins to approach zero rapidly under the influence of near-linear, critically damped proportional plus derivative control. Ideally, continued braking at -10.5 N-m would have brought the system to rest at the goal in an additional 6.4 ms. Instead, the P-D controller brings the system to rest in 10.3 ms. with a slight overshoot. Subsequently, the P-D controller corrects for the overshoot.

A perfect bang-bang controller with torque saturation at 13 N-m would have performed the 3 rad step in a time of 0.208 seconds, with a torque transition at 0.104 seconds. A perfect bang-bang controller with a positive torque limit of 13 N-m and a negative torque limit of -10.5 N-m would have performed the transition in 0.2209 seconds, with a torque reversal at t = 0.0982 seconds. The sliding-mode controller reached its maximum position (with slight overshoot) at 0.2195 seconds, with a torque transition at 0.0985 seconds.

The sliding-mode controller performed nearly as well as the theoretically perfect controller. For a comparison with conventional linear control, two more control routines were implemented and measured: a proportional plus derivative control with torque saturation, and an unsaturated proportional plus derivative control. The same hardv are was used for these experiments, and the sampling and data acquisition rate of 1.5 kHz was preserved.

Performance of the first alternative controller, saturated P-D control, is shown in figures 7 and 8. In this example, the same values for K and b were used as

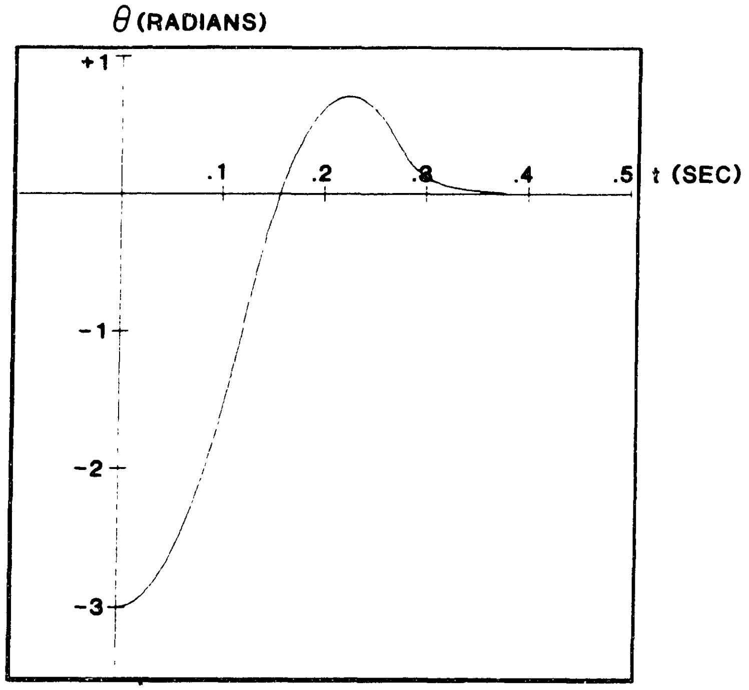

Figure 7: Saturated P-D Control: Angle vs Time

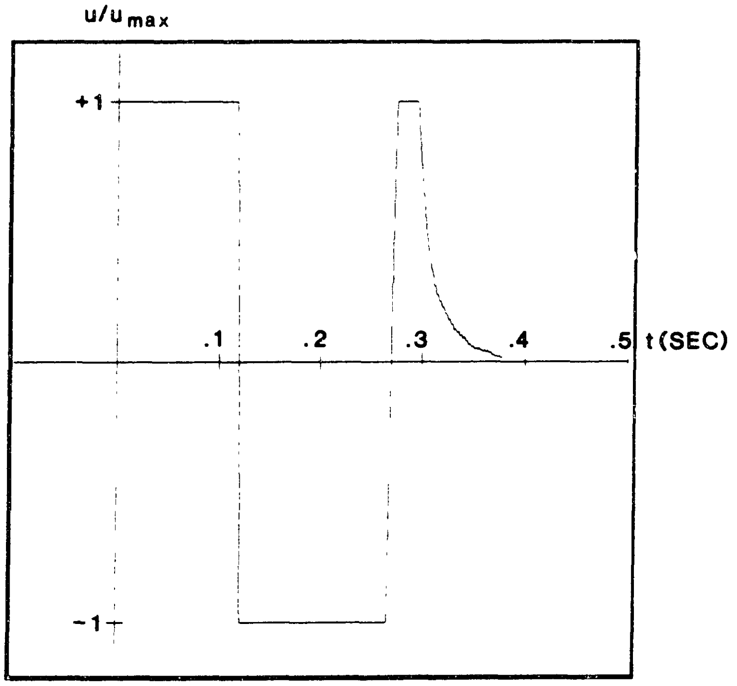

Figure 8: Saturated P-D Control: Normalized Torque vs Time

in the near-linear regime of the sliding-mode controller: 185 N-m/rad and 5.9 N-m/r/s, respectively. If torque saturation were not present, then the system would have exhibited critical damping with a natural frequency of 10Hz. However, with torque saturation at 13 N-m, the proportional term alone causes saturation for a position error of only 0.07 radians, which is only 2.3% of the example 3 radian step command. Control was implemented as though saturation did not occur, and the resulting torque commands were saturated by the physical limits of the amplifier.

Position vs time is shown in figure 7. Although linear theory predicts zero overshoot, violation of linear assumptions due to torque saturation resulted in an overshoot of 0.71 radians, or 24% of the 3 rad step command.

The torque history, shown in figure 8, clearly indicates the influence of torque saturation. The torque profile starts with full positive saturation for 0.220 seconds. Since theoretically perfect control would exert maximum torque for only 0.104 seconds, the saturated P-D controller has accelerated the link well beyond the point at which overshoot can be prevented. At 0.220 seconds, the controller rapidly switches to negative torque saturation. Although maximum braking is applied, the inertia passes through the goal position with a velocity of 22 rad/sec. At 0.270 seconds, the controller switches torque again to positive saturation. The valid linear region is entered only after 0.30 seconds, at which time the torque command falls below saturation and continues to decline monotonically to zero under valid linear control. No more oscillations are observed after entering the linear control region.

The second alternative controller is also proportional plus derivative. However, the gains K and b were reduced to 4.33 N-m/rad and 0.90 N-m/(rad/s), respectively. This choice of gains results in critical damping with a natural frequency of about 1.5 Hz. These choices for K and b are the absolute best linear gains which: 1) preserve linear assumptions (no torque saturation for a 3 rad step command); 2) guarantee zero overshoot; 3) satisfy the previous conditions

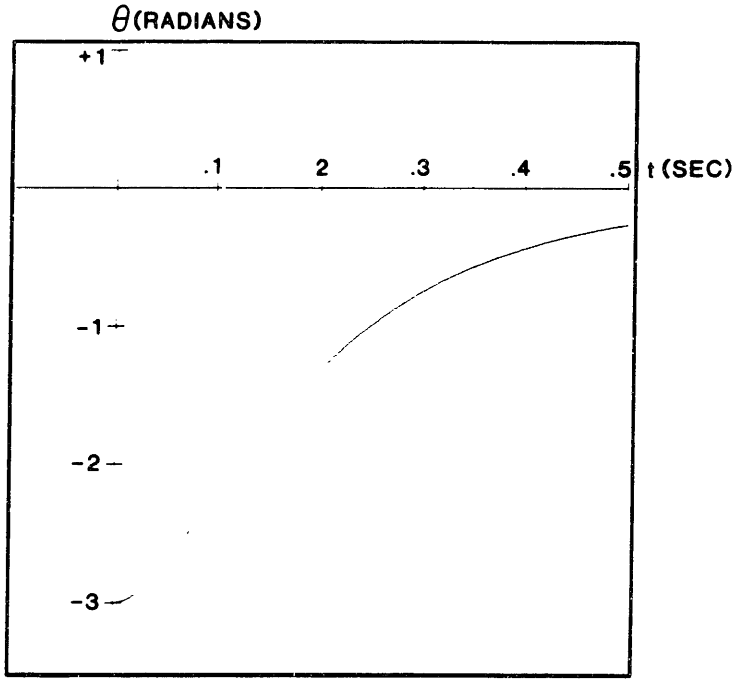

Figure 9: Unsaturated P-D Control: Angle vs Time

in minimum time under P-D linear control. The results are shown in figures 9 and 10.

Figure 10 shows the torque history under unsaturated linear control. Initially, the torque command is the maximum feasible torque of 13 N-m. After time zero, the torque continues to fall, and thus remains within the valid linear control region throughout the trajectory. Any higher value of K would cause torque saturation, which invalidates the linear analysis used to select the controller.

The resulting position history is plotted in figure 9. No overshoot occurs, but after 0.50 seconds, the link is still 0.264 radians from its goal. The gains which preserve linearity are so low that the torque commands fall to within the magnitude of Coulomb friction after 0.37 seconds, at which time the link is still 0.51 radians from its goal. Under linear control, the response is slow, and the regulation is poor.

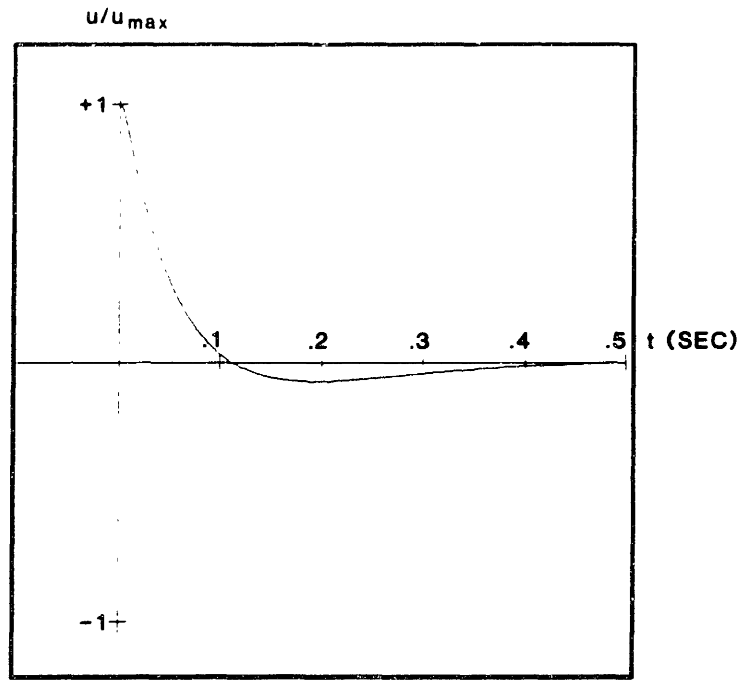

Figure 10: Unsaturated P-D Control: Normalized Torque vs Time

§9. Summary and Conclusions

A sliding-mode controller has been presented which imitates time-optimal control, but does so robustly. The limitations of bang-bang control, sensitivity to parametric errors and instability resulting from unmodeled dynamics, have been alleviated in the proposed scheme. The controller is tolerant of control effort saturation, parametric uncertainty, and unmodeled high-frequency dynamics. A design procedure was presented for smoothly blending bang-bang control into linear control with small overshoot. Global stability proofs were presented for all regimes of operation: saturation, unsaturated sliding-mode regulation, and (nearly) linear control about the goal state.

The analysis presented here has been restricted to a frictionless, second-order system. Generalization to higher dimensions should follow analogous deductions and proofs, but such generalization is not attempted here.

The sliding-mode controller was implemented on a planar, two-link robot.

Test results for link 1 for a large (3 rad) step command demonstrated a move time which was only 6% slower than theoretically perfect control, and exhibited an overshoot of only 0.06%. Torque histories resulting from the sliding-mode controller approximated the bang-bang control law derived by time-optimal control. In the decelerating half of the control history, high-frequency torque variations were observed. Control smoothing through the use of saturation functions should have eliminated such chatter. The source of the remaining chatter was not determined; discretization of the position measurments (via optical encoders) is a suspected contributor. The torque chatter damped out during deceleration, and did not lead to instability or poor control performance in the present case. However, the source of chatter should be investigated for future extensions of the theory.

Linear proportional plus derivative controllers were tested for comparison with the sliding-mode optimal controller. The linear controllers exhibited large overshoot when torque-limited, or very slow response for the best unsaturated linear control. For an allowed overshoot equal to that of the sliding-mode controller, the best linear controller is approximately four times slower than the optimal controller for the chosen 3 radian step command. For longer moves, the performance comparison between the optimal controller and the best linear controller will be more dramatic. On the other hand, as the commanded move approaches zero, the optimal controller will degenerate to a linear controller; no advantage in movement speed will then be observed.

The key feature of the new control law is that feedback control is applied to drive a system to imitate some desirable and feasible set of dynamics. The desired dynamics in the present case were specified as a state constraint corresponding to motion along the switching curve of a time-optimal control law. The state constraint is expressed as a relation between velocity and position. This is in contrast to conventional approaches in which a torque history or desired position trajectory is pre-computed. The success of the present approach suggests application to other types of control in which some desired set of dynamics is to be achieved. Potential application areas include balancing, walking, catching, control of two-handed operations and operations involving intermittent contact. In each case, some desired set of dynamics should be deduced, rather than any a priori position, velocity or force trajectory, and the sliding-mode technique should be adapted to enforce the chosen dynamic relationship.

CHAPTER IV

CONFIGURATION SPACE TRANSFORMS FOR OBSTACLE AVOIDANCE

§1. Introduction

In the previous chapters, first design and then control for a very high speed two degree of freedom, planar manipulator were described. The high-speed capabilities accentuate the danger of collision with possible obstacles within reach of the robot. The problem of collision avoidance involves three major issues which, it will be shown, can be treated separately and implemented as separate but coordinated modules. First, there is the problem of observation, computation and representation of the obstacles within the robot's reach. Second, there is the servo-level issue of reconciling very high-speed robot motion with guaranteed safety in the presence of reachable obstacles. Finally, there is the planning-level issue of finding a path among the obstacles which successfully moves the robot to a goal location without collisions. These three issues will be treated, respectively, in the following three chapters.

In this chapter, obstacle computation and representation will be treated. Objectives of the obstacle computation/representation scheme are, in short, speed and generality.

Any obstacle-based computations must be fast to accommodate dynamic environments, (cases in which the obstacles are not all stationary). Although "fast" is a relative term, a heuristic qualification of a fast computation can be described in terms of the speeds of the moving obstacles. A "fast" computation is one which performs obstacle-based computations rapidly enough to competently track moving obstacles. Another aspect of the speed issue is that the representation scheme should permit efficient use of the obstacle computations. The representation should be efficient enough that access time should be transparent, and that further transformations for use by the robot controller and path planner should be minimal. The maximum speed of the robot should be limited only by physical constraints of the electromechanical design, not by information access and processing delays.

Generality is the second major requirement of an obstacle computation and representation approach. The scheme should be extensible to arbitrarily complex environments, including obstacles and robots of arbitrary shape, very cluttered environments, and high-dimensional cases. Further, the ideal approach would provide arbitrarily high resolution. The preceding virtues should not compromise computation speed.

In this chapter, a fast and general approach is presented for obstacle transformations and representation. The technique is detailed for the 2 d.o.f manipulator described in chapter 2. The procedure uses discretized "configuration space" Both static and dynamic obstacles are included in a common representation. Dynamic obstacles are updated in configuration space at very high rates. The chosen representation, which has been implemented in 2-D, effectively handles arbitrarily complex 2-D environments. The chosen space (joint space) is ideal for high-speed access by both the servo controller and the planner. The techniques demonstrated in 2-D are generalizable to higher dimensions, albeit with penalties in storage requirements and computation times.

§2. Configuration Space Description of Obstacles

The manifold objectives of the obstacle representation scheme comprise competing criteria. A good compromise among the conflicting requirements will be described here in terms of "configuration space" [17,25,26,39]. In configuration space, an obstacle is represented with respect to a particular robot in terms of forbidden regions in the joint space of the robot. Each point in the joint space of a robot corresponds to a unique pose, or "configuration" of the robot. Not all points in joint space can necessarily be reached in practice, since physical joint limits typically impose bounds on attainable joint positions. Joint limits, though, are merely feasibility constraints; every point in joint space is mathematically well defined as a robot pose. There are no complications with redundant degree-of-freedom kinematics, multiple solutions, undefined states or singularities, as there are in a task-space representation [20]. On the other hand, describing obstacles in terms of robot joint angles can be non-intuitive and difficult to compute.

Obstacles in configuration space appear as forbidden regions of joint values. For a 2 d.o.f. robot, the regions are areas; in three dimensions, the regions are volumes; in higher dimensions the regions are hyper-volumes. Edges (or surfaces or hypersurfaces) of the forbidden regions correspond to poses of the robot for which some point on the physical envelope of the robot is in contact with some point on the physical envelope of an obstacle. In this representation, joint limits of the robot can be expressed in the same manner as other obstacles. A joint limit "obstacle" in configuration space is the half plane (half space, half hyperspace) bounded by the line (plane, hyperplane) defined by the one-dimensional constraint: joint variable = joint limit. In addition to encountering individual joint limits, a robot may be capable of striking its own body. Joint angles which correspond to such a condition can also be expressed in terms of forbidden regions in joint space. The conceptual visualization of forbidden regions in joint space has suggested the alternative name, "image space" [28,29,30].

The simplest example of an image space is that of a point automaton moving among a set of obstacles. In this case, image space is the same as the space in which the automaton moves, and the forbidden regions are identically the physical volumes of the obstacles. A slightly more complex case is that of a spherical automaton moving among obstacles. The corresponding configuration space map looks much like the original space of obstacles, only the boundaries of the obstacles in configuration space are "grown" by an amount equal to the radius of the automaton. (See, e.g., [25,26,49]).

For a Cartesian manipulator, the configuration space map is similar to that of the spherical automaton, in that the configuration space obstacles are typically identifiable as corresponding to physical obstacles. The configuration space obstacles are "grown" by an amount depending on the cross-sectional dimensions of the robot links. In addition, however, interference with the robot links can result in more dramatic "outgrowths" of the physical obstacle boundaries in configuration space. Nonetheless, configuration space maps are relatively easy to compute for Cartesian robots. Articulated robots, on the other hand, have configuration space maps which can be both difficult to compute and difficult to interpret.

In the use of configuration space described here, joint space is discretized into minimum resolution areas which, extending the "image space" analogy, will be referred to as "pixels" (or "voxels" in 3-D). Implementation has been limited to two dimensions, conforming to the joint space dimension of the experimental robot described in chapter 2. A high-resolution discretization of joint space becomes inefficient in storage and computation as the number of dimensions grows. However, it will be shown that two dimensions is entirely practical with current technology, and that three dimensions is not unrealistic. Although most industrial robots have four to six degrees of freedom, a configuration sub-space map of only two or three dimensions is arguably adequate.

An important practical example is the popular SCARA type robot, [13,31], which has four degrees of freedom. If the gripper is not carrying large tools or large parts, then the fourth degree of freedom (wrist roll) may be ignored by modeling the gripper with a bounding cylinder. Then, configuration-space obstacles may be adequately described in terms of the three remaining joint coordinates.

For robots with higher degrees of freedom, the same approximation is often valid. For articulated robots as well as Cartesian robots, typical designs use three long links which essentially determine the position of the end effector. A "spherical wrist" [37,38] is often employed, which controls orientation of the end effector. With a spherical wrist, at least two of the link lengths are zero. If the length of the end effector is reasonably short, or if a gripper at the end is not handling large parts, then the wrist and end-effector may be approximated by a reasonably small bounding sphere centered at the intersection of the wrist axes. The bounding sphere approximation reduces a six-dimensional configuration space to only 3 dimensions. Even in cases where the bounding-sphere approximation seems overly conservative, its use may be justified by the tremendous savings in computation time and storage space resulting from reducing the problem dimension from 6 to 3.

§3. Properties of Configuration Space Obstacles

In this section, some general observations will be made about configuration space obstacles with particular reference to the robot described in chapter 2. The chosen joint space for this robot is absolute joint angles, not relative joint angles. That is, if joint 1 is moved while joint 2 is kept constant, the angle of the elbow will change. The center of gravity of link 2 would translate as link 1 moves, but link 2 would not rotate with respect to a Newtonian frame of reference. Since this is a particularly convenient coordinate system for robot control, it has been chosen also to be the coordinate system for configuration space.

Since the robot under consideration is capable of continuous rotation of both links, its configuration space representation must accommodate periodicity in both dimensions. A representation which satisfies the 2-D periodicity requirements is the surface of a torus. The surface of a torus is two dimensional, and a coordinate system can be defined on this surface which is periodic in both dimensions. For further description of this representation, see [28,29,30]. Configuration space maps presented herein appear to be planar figures, though they should be understood to imply periodicity: the top edge joins the bottom edge and the right edge joins the left edge of each map.

The most fundamental obstacle in configuration space is a point obstacle. In fact, point obstacles are not physical; a dimensionless "obstacle" could hardly pose much of a threat. Point obstacles are well-defined mathematically, though, and are highly useful in computing configuration space representations of real obstacles.

Although a point obstacle has zero area (or volume), its configuration space image may consume substantial area (or volume, or hypervolume). A point obstacle can map into configuration space as multiple, disconnected forbidden regions, as a single region, or as nothing at all. Forbidden regions are by no means guaranteed to be convex. Separate obstacles in task space, which may be spaced far apart, can produce forbidden regions which overlap dramatically in configuration space.

Some representative examples of point obstacle transformations for the 2-D planar robot are shown in figures 1 through 6. The figures shown were generated physically by tracing about a small (2mm) diameter pin placed within reach of the robot. For each location of the pin, the robot was moved manually while maintaining contact between the pin and the robot. The joint angles of the robot were recorded by the control computer during this operation, and plotted out afterwards. Each point obtained in this manner corresponds to a point on the edge of a forbidden region in configuration space. The edge points form one or more closed contours which enclose forbidden regions in configuration space. Figures 1 through 6 correspond to pin placements along the x-axis of the robot's workspace; the x-axis was defined (arbitrarily) as an axis along which to define the origin of joint space. When the arm is in its pose corresponding to 0 angle of both joints, it is at full extension, outstretched along the positive x-axis of its workspace. The pin placements of figures 1 through 6 correspond to x placements of 45 cm, 40 cm, 35 cm, 28.5 cm, 20 cm, and 10 cm, respectively. Any point obstacle beyond 51.5 cm has a null mapping in configuration space, since it is

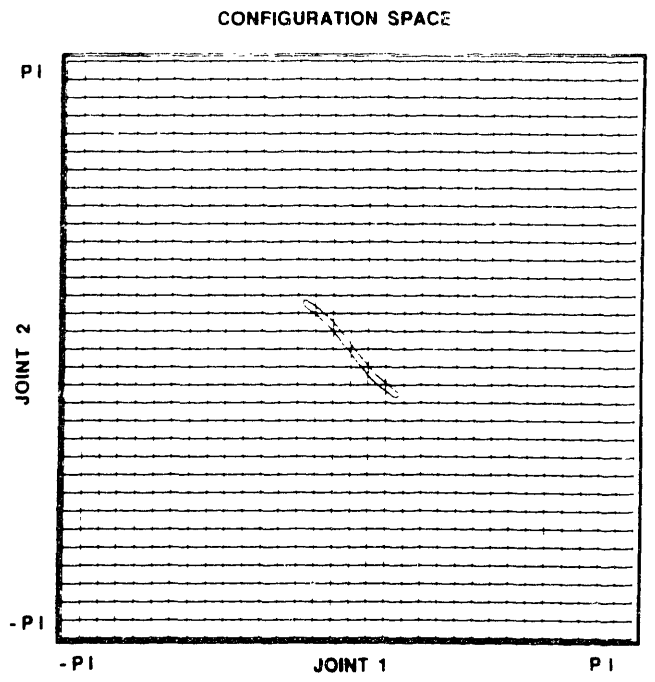

Figure 1: Configuration Space Point Obstacle at 45cm

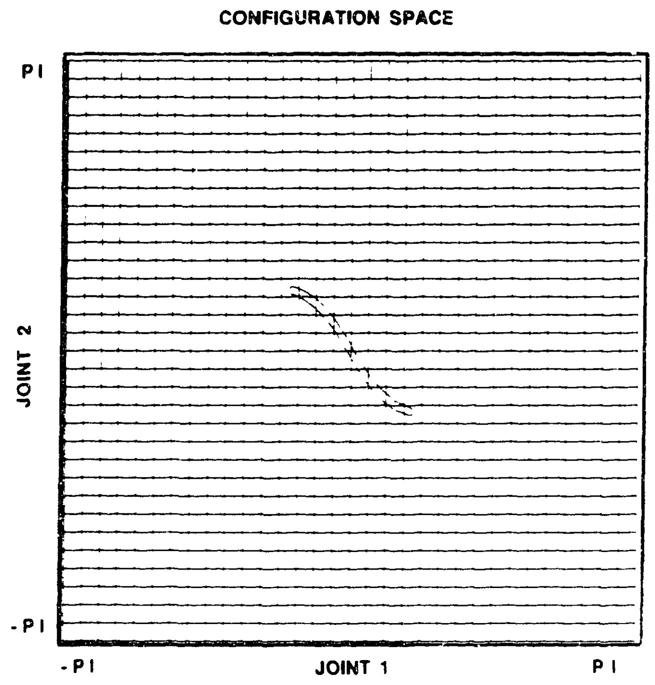

Figure 2: Configuration Space Point Obstacle at 40cm

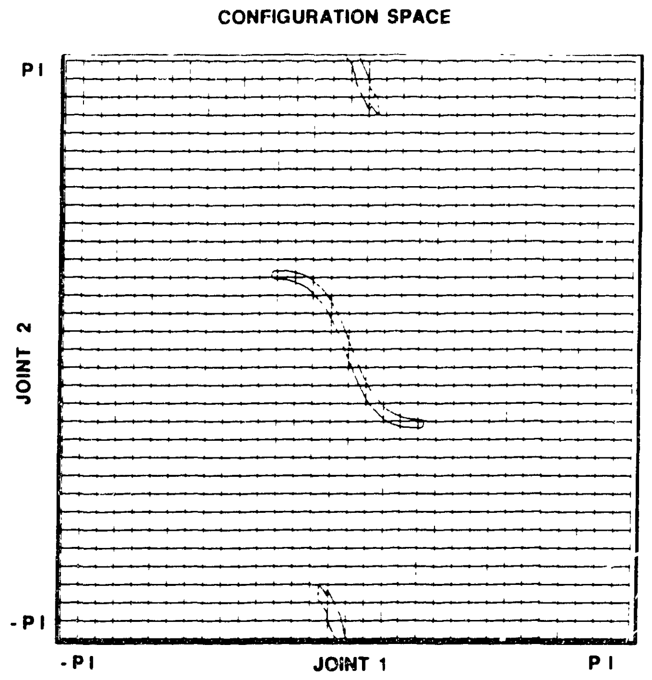

Figure 3: Configuration Space Point Obstacle at 35cm

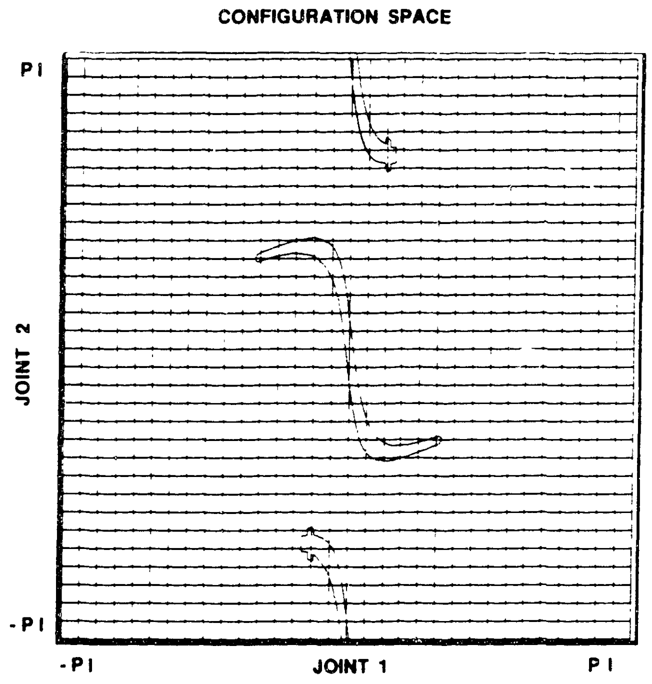

Figure 4: Configuration Space Point Obstacle at 28.5cm

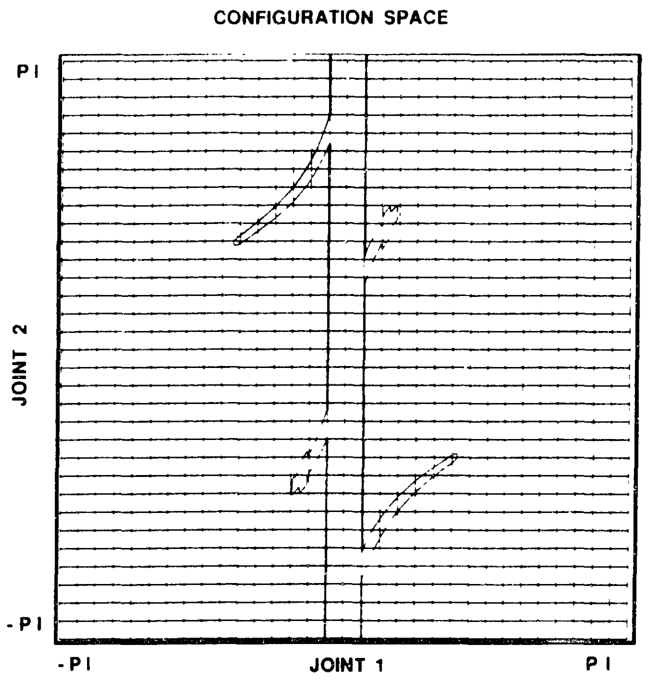

Figure 5: Configuration Space Point Obstacle at 20cm

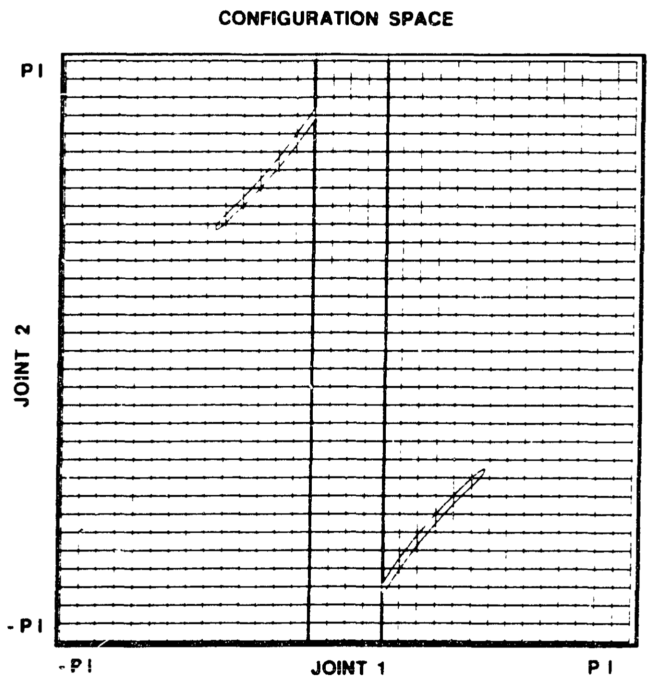

Figure 6: Configuration Space Point Obstacle at 10cm

impossible for the robot to reach the obstacle. Any obstacle within 3.8 cm maps onto the complete configuration space, since such points are within the body of the robot, and no combination of joint angles can prevent the overlap.

In figure 1, the configuration space map is shown for a point obstacle near the extreme reach of the robot. Although the ideal point obstacle has zero area in task space, (and the actual pin used has negligible area), the forbidden area in configuration space is certainly non-negligible. Essentially, the point obstacle is "grown" by an amount related to the width of the robot arm. The locus of points comprising the contour in figure 1 is the locus of joint angles for which some part of the surface of the robot is in contact with the pin. Equivalently, this means that the point x= 45cm, y=0 lies on the boundary of the area corresponding to the image of the robot projected on the x, y plane. The area enclosed by the contour in figure 1 corresponds to all positions of the robot for which the projected profile of the robot contains the point y=0, x= 45 cm. All possible collisions between the robot and the pin at the chosen location are represented by a single closed region in configuration space. The only possible collisions with the chosen point occur within the robot's tip, or somewhere within the envelope of link 2 beyond 20 cm from link 2's revolute joint. It is not possible for any part of the "tail" of link 2, the section supporting the countermass, to intersect the specified obstacle. The obstacle is also beyond the reach of link 1.

Figure 2 was obtained in the same manner, with the "point" obstacle moved to 40 cm on the x axis. The resulting configuration space map is conceptually similar to that of figure 1, though broader and distorted. Still, only a single closed region is formed. The obstacle at this location is beyond the reach of the tail of link 2.

In figure 3, the obstacle has been moved to x = 35cm, which is within reach of the tail of link2. It is still beyond the reach of link 1, though. The closed region containing the origin corresponds to collisions between the "front" of link 2 and the point obstacle. It is similar to the obstacle regions in figures 1 and 2.

In addition, though there is a new region which corresponds to the joint angles of the robot for which the tail of link 2 contains the point obstacle. The locus of edge points for tail collisions does, in fact, form a closed contour, since the top of the graph is periodic with the bottom of the graph. The two sections at the top and bottom of the graph are two halves of a single region. Thus there are only two separate, closed regions in the map of figure 3; these regions will be referred to in the following as "link2-front" and "link2-rear" configuration space obstacles.

Figure 4 shows the point obstacle at 28.5cm, which is almost within reach of the elbow. Both the link2-front and link2-rear obstacles are distorted from their respective images in figure 3. The edges of the obstacle regions have slopes opposite that of centers. The link2-rear obstacle clearly shows the outline of the countermass on the tail of link2.

Figure 5 appears quite different from the preceeding cases. In this figure, the point obstacle is at 20cm, which is within reach of both the front and rear of link 2, as well as within reach of link 1. Actually, the link2-front and link2-rear collision regions are still present, but there is an additional obstacle region describing collisions with link1. This additional region excludes a vertical stripe in configuration space, which overlaps both the link2-front and link2-rear regions. The new region will be called a "link1" configuration space obstacle. The borders of the link1 region appear in figure 4 as two vertical lines. The area contained between these lines comprises a band of forbidden poses; the top of the band wraps around to join the bottom of the band to form a continuous strip. The edges of a link1 obstacle are vertical, since contact between link1 and an obstacle is independent of the position of more distal links (link2). Although link1 obstacles are compatible with the 2-D configuration space representation, link1 collisions could be fully described by a one-dimensional representation. This is a general principal. For serial links numbered sequentially from the ground to the most distal link, link "i" obstacles require an i-dimensional configuration space representation. Links closer to ground can be mapped fully into lower dimensional

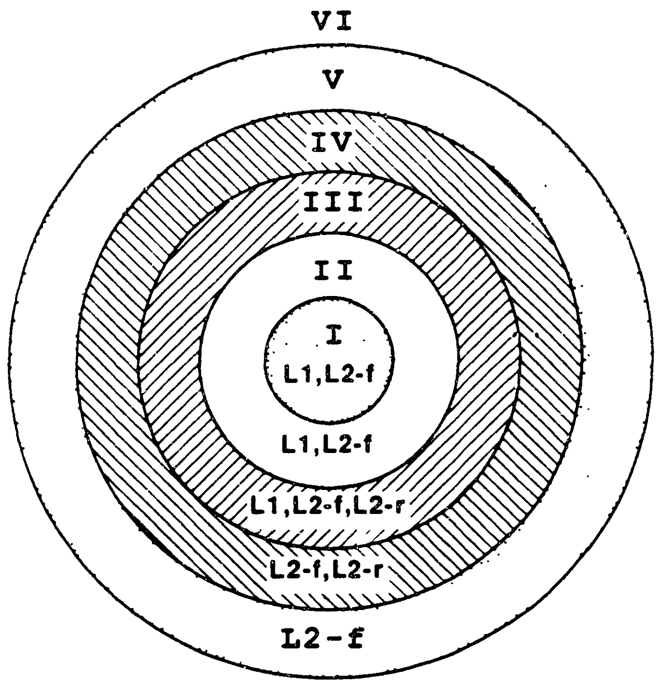

Figure 7: Configuration Space Obstacle Types vs Radius

obstacle representations.

In figure 6, the point obstacle is at x = 10cm. At this location, the obstacle is within reach of the front of link2, and is within reach of link 1, but is not within reach of the rear of link 2. At its closer proximity, the obstacle excludes a broader stripe in configuration space due to the link1 obstacle contribution. The link2-front contribution is still present, though its connectedness is obscured by the overlayed link1 obstacle.

The regions of existence of the three obstacle types are illustrated in figure 7. In figure 7, point obstacles in configuration space are classified as falling within six regions. Region I includes all points within 3.8cm of the axis of link 1. Points within this region fill configuration space entirely. Actually, it is only the link1 type map which fills all of configuration space; since link1 is 3.8cm wide, collisions between link 1 and the obstacle point occur for all angles of link 1, regardless of the angle of link 2. Obstacle points within this region also may collide with the front of link 2, though the link2-front obstacle map does not fill all of configuration space.

In region II, which contains points lying between radii of 3.8 cm and 12 cm, configuration space is no longer completely filled. Both link1 and link2-front type obstacles appear in configuration space for points in this region.

For obstacle points in region III, which lies between radii of 12 cm and 27.5 cm, all three types of obstacles appear in configuration space: link1, link2-front, and link2-rear. In region IV, between 27.5 cm and 38 cm, link1 obstacles disappear, and link2-front and link2-rear obstacles are present as completely separate regions. In region V, between 38 cm and 51.5 cm, only link2-front obstacles are present. Finally, region VI, which contains all obstacle points beyond a radius of 51.5 cm, generates no forbidden regions at all in configuration space.

It will be useful to distinguish the separate obstacle types and their respective regions in computing configuration space transforms.

§4. Constructive Geometry for Configuration Space Images

Although the mapping from points in task space into forbidden regions in configuration space seems complex and ill-behaved, some powerful mathematical properties of the mapping can be deduced.

First, a general property of the mapping of sets from task space to sets in configuration space can be stated as follows: For sets O_1 and O_2 in task space, the transformation $\mathcal{T}$ which maps task space points into configuration space images satisfies the identity:

$$\mathcal{T}(O_1 \cup O_2) = \mathcal{T}(O_1) \cup \mathcal{T}(O_2) \tag{IV.1}$$

That is, the transformation of a union is equivalent to the union of the transformations. This property follows from the definition of transformations of individual points from task space to configuration space. A point x in task space, $\mathbf{X}$, maps onto an "image" set, I_y of points y in configuration space, $\mathbf{Y}$ via the transformation $\mathcal{T}$: $I_y = \mathcal{T}(x)$. Alternatively, a point $y \equiv (\theta_1, \theta_2)$ in joint space defines an inverse image in task space, denoted by $\mathrm{R}_x(y)$. The inverse image is simply the set of all points in task space which lie within the physical envelope of the robot when the robot is in position y. The mapping from task space into configuration space is actually defined in terms of this inverse map. That is, a point y is an element of $I_y = \mathcal{T}(x)$ if and only if x is an element of $\mathrm{R}_x(y)$.

The complement of $I_y(x)$, denoted by $\mathrm{C}I_y(x)$, is the set of all points in $\mathbf{Y}$ which are not in $I_y(x)$. The complement of $I_y(x)$ is "free space" since for every point y in $\mathrm{C}I_y(x)$, the point x in $\mathbf{X}$ is not contained in $\mathrm{R}_x(y)$. That is, the pose y does not result in a collision between the robot and the point obstacle, x. Transformations of sets of points in $\mathbf{X}$ can now be defined in terms of transformations of points in $\mathbf{X}$.

The definition of the transformation of a set of points in $\mathbf{X}$, S_x, will be written with the same notation: $I_y(S_x) = \mathcal{T}(S_x)$. Set transformations will be defined based on preservation of the notion of free space. Specifically, the complement of $I_y(S_x)$, $\mathrm{C}I_y(S_x)$, consists of all the points y for which the intersection of $\mathrm{R}_x(y)$ and S_x is empty. In other words, free space in $\mathbf{Y}$ means that the robot does not collide with any of the points in S_x. Free space in $\mathbf{Y}$ is defined with respect to point x_1 in $\mathbf{X}$ by the complement of the point transformation, $\mathrm{C}\mathcal{T}(x_1)$. Free space in $\mathbf{Y}$ with respect to x_2 in $\mathbf{X}$ is $\mathrm{C}\mathcal{T}(x_2)$. A point y in $\mathbf{Y}$ satisfies the criterion for being in the free space of $\mathbf{Y}$ with respect to the set of points $S_x = x_1, x_2$ if and only if y is an element of $\mathrm{C}\mathcal{T}(x_1)$ and y is an element of $\mathrm{C}\mathcal{T}(x_2)$, i.e., y lies within the free space corresponding to each of the points. Based on the definition of free space for set transformations, the following is true:

$$\mathrm{C}\mathcal{T}(S_x) = \mathrm{C}\mathcal{T}(x_1) \cap \mathrm{C}\mathcal{T}(x_2) \qquad \text{(IV.2)}$$

That is, the free space resulting from of the set of points x_1 and x_2 is the intersection of the free space corresponding to x_1 and the free space corresponding to x_2.

Taking the complement of equation IV.2 yields the relation:

$$\mathcal{T}(x_1 \cup x_2) = \mathcal{T}(x_1) \cup \mathcal{T}(x_2) \tag{IV.3}$$

Equation IV.3 expresses an identity for the transformation of a set consisting of two points. The identity may be applied recursively to generalize it to arbitrary sets in $\mathbf{X}$, which establishes the set relation of equation IV.1. There is no restriction on the type of sets in $\mathbf{X}$ which satisfy equation IV.1; e.g. convexity, closure, connectedness, etc.

The above approach to computing configuration space obstacles is rigorous, but inefficient. A dramatic efficiency improvement may be obtained by realizing that it is only necessary to map points on the boundary (surface) of the physical obstacle into configuration space. This is a consequence of equation IV.1. A lemma of equation IV.1 can be stated as follows: a subset of the domain of any configuration space transformation maps into a subset of the range of the original transformation. Consequently, a physical obstacle which is fully contained within a larger obstacle has a configuration space map which is fully contained within the configuration space map of the larger obstacle. A limiting case of this property is that the configuration space image of any obstacle is contained within the the image of the obstacle's boundary. Therefore, it is only necessary to transform points on the edge (surface) of an obstacle to create a map which fully contains all collision states of the robot for the entire area (volume) of the obstacle. In practice, a fairly coarsely spaced, finite subset of boundary points may be transformed to obtain a good approximation of the bounding shell of the complete configuration space map. Roughly, the spacing of sample points on the physical obstacle's boundary should be somewhat smaller than the width of the smallest robot link.

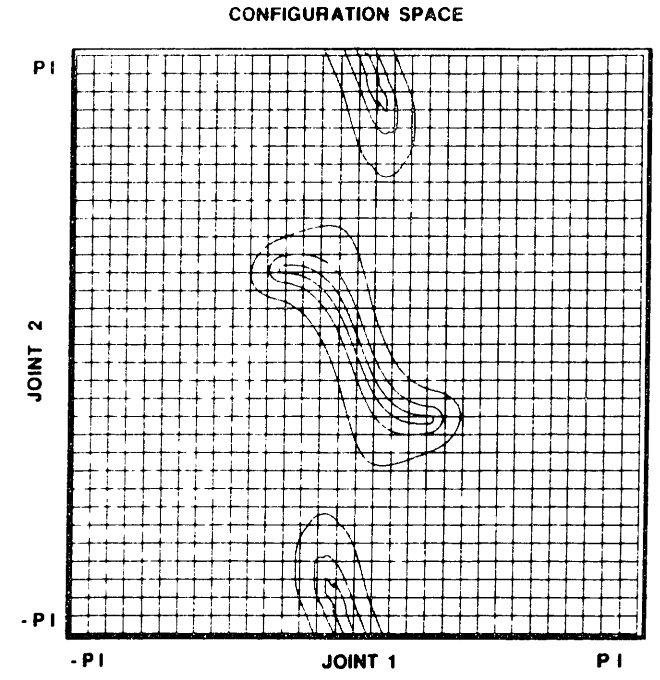

Figure 8: Concentric Obstacles at 35cm: C-Space Representation

Since arbitrary obstacles can be constructed in configuration space through the union of point transformations, it is only necessary to be able to perform point transformations efficiently. Further improvements in speed, however, can be obtained by introducing additional elements for constructing configuration space geometries. In particular, circles (or spheres) are convenient building blocks.

Figure 8 shows three superimposed outlines corresponding to the configuration space link2-front forbidden regions for a point (actually a 0.030" diameter pin), a 2" diameter circle, and a 5.75" diameter circle. Each of the obstacles was centered about the position x= 35 cm, y = 0. As deduced, the larger circles in task space, which completely enclose the smaller circles, map onto configuration space regions which completely enclose those corresponding to physically smaller obstacles. Transformations of circles into configuration space may be used to construct configurations space images of more complex obstacles. An obstacle in task space may be modelled as the union of disks of various diameters. The resulting configuration space transform is the union of the transforms of the individual disks.

In practice, the composite model in task space is permitted to contain holes, since only the shell of the task space obstacle requires transformation. As long as the complete boundary of the task space obstacle is contained within one or more of the model elements, the union of the transformations of the model elements will produce a configuration space mapping which completely encloses the exact configuration space mapping of the actual obstacle.

Modeling elements may be added to task space at will, and the corresponding joint space image of each element may be superimposed on the configuration space map. Subtraction of images from configuration space is not as easily performed, however, since:

$$\mathcal{T}(O_1 - (O_1 \cap O_2)) \neq \mathcal{T}(O_1) - \mathcal{T}(O_1 \cap O_2) \qquad \text{(IV.4)}$$

An additional valuable constructive geometry technique is the use of "swept" primitives. For example, a line in task space may be thought of as a set of points, or a single point swept along the line. A stripe (cylinder) in **X** may be modelled as a swept circle (sphere). The advantage of this viewpoint is that instead of superimposing images in **Y** from a high density of points in **X**, images in **Y** may be constructed from a "blend" of images of a sparse set of points in **X**. Blending configuration space images is an interpolation technique which is justified only under the condition of functional continuity.

The concept of continuity of a function requires the definition of a metric. An acceptable metric for the configuration space transformation may be defined by:

$$d_A(\mathcal{T}(x_1), \mathcal{T}(x_2)) \equiv Area(\mathcal{T}(x_1)) + Area(\mathcal{T}(x_2) - Area(\mathcal{T}(x_1) \cap \mathcal{T}(x_2)) \qquad \text{(IV.5)}$$

where "Area" is understood to mean the surface integral (or volume integral in higher dimensions) of the areas (volumes) contained in the closed region(s) of the argument (configuration space images). A suitable metric on task space is the Euclidean norm (minimum distance between points).

To prove continuity of the mapping $\mathcal{T}(x)$ about a point x_0, it is sufficient to show that given any $\epsilon > 0$, there exists a $\delta > 0$ such that $d_A(\mathcal{T}(x), \mathcal{T}(x_o)) < \epsilon$ whenever $||x - x_0|| < \delta$. For the chosen metric, the preceding statement is equivalent to saying that as a point x approaches the point x_0 the image of x in $\mathbf{Y}, \mathbf{I} = \mathcal{T}(\mathbf{x})$, becomes increasingly similar to the image $\mathcal{T}(x_0)$. This condition is not rigorously proven here, but has been empirically observed to be true, at least within each of the regions demarcated in figure 7. Continuity of the configuration space transformation does not hold across the boundaries of the regions in figure 7.

A net configuration space mapping for the 2 d.o.f. planar arm may be thought of as a composite of the three individual mappings, each of which is continuous within specified ranges. In particular, link1 obstacles are well behaved within regions II and III of figure 7. Incremental perturbations of obstacle point positions within these regions result in incremental variations of the configuration space maps. Link1 obstacle mappings are continuous within this range, and may be interpolated, but link1 obstacles may not be interpolated across the inner boundary of region II or across the outer boundary of region III.

Link2-rear obstacles are also generated by a continuous mapping from points in task space to a single closed region in joint space. The region of continuity includes the regions III and IV in figure 7, excluding the boundaries between regions II and III and between regions IV and V. Link2-front obstacles have a continuous mapping everywhere within the boundary of region VI.

Having justified constructive geometry based on circles (spheres), and having justified the use of numerical interpolation for configuration space transforms, a fast configuration space transformation computation process can now be described.

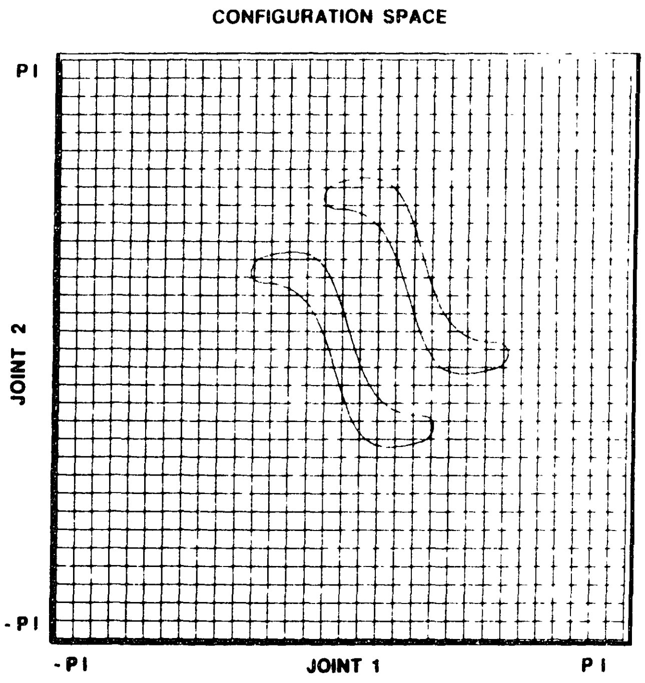

Figure 9: 2" Obstacles at 0 deg and 45 deg; C-Space

§5. Configuration Space Transformations for Circular Obstacles

An implementation of fast transformations is described here. It has been realized for 2" diameter circles, i.e. cylindrical obstacles in the workspace of the planar robot. The same technique is immediately applicable to circular obstacles of any diameter.

For fast transformations of circular sets in task space, it is most efficient to pass through an intermediate frame of polar coordinates. Pcint obstacles in task space which lie at the same radius from the axis of joint 1 rotation have geometrically identical images in configuration space. A non-zero polar angle causes a pure shift of the forbidden region in joint space, without scaling or distorting the image.

Figure 9 shows the link2-front configuration space representation of two 2" obstacles at the same radius (33cm) but at different polar angles (0 deg and +45 deg). The images in configuration space differ only by a pure translation. The obstacle on the x-axis (polar angle of 0 deg) has odd symmetry about the

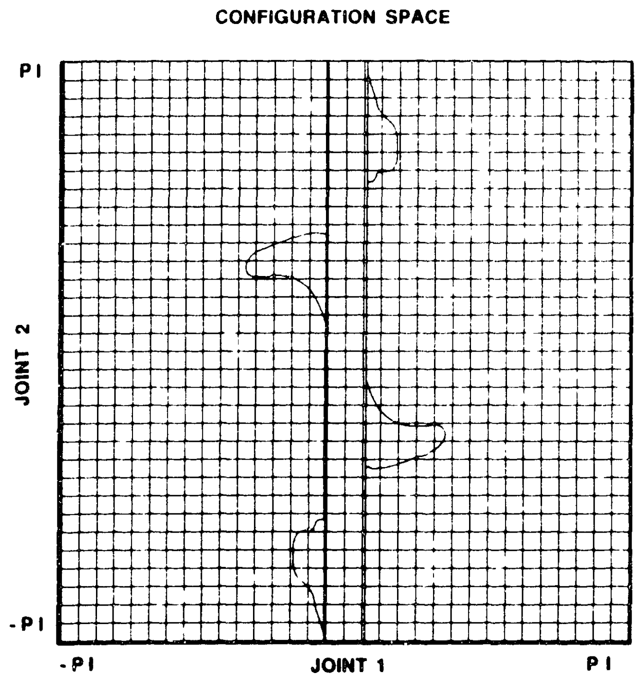

Figure 10: 2" Obstacle at 30cm; C-Space

origin ($\theta_1 = 0, \theta_2 = 0$). The 2" obstacle at (r, α) = (33 cm, +45 deg) has a corresponding configuration space image with odd symmetry about the point (θ_1, θ_2) = (+45 deg, +45 deg) in joint space. Thus, the transformation of points from polar coordinates in task space to forbidden regions in joint space is a linear transformation in polar angle α, and nonlinear only in radius, r. Consequently, to perform efficient transformations of points from task space to configuration space, it is sufficient to know how to perform the efficient transformations for points along the positive x axis only.

A fast transformation method for circles along the positive x-axis is described here in terms of bounding polygons. Examples of exact configuration space images for 2" diameter obstacles are shown in figures 10 and 11. In figure 10, the obstacle is centered at $x = 30$ cm , $y = 0$. All three obstacle types are apparent: link2-front, link2-rear and link1. In figure 11, only link2-front obstacles are graphed. Obstacle transforms for three different radial placements are superimposed.

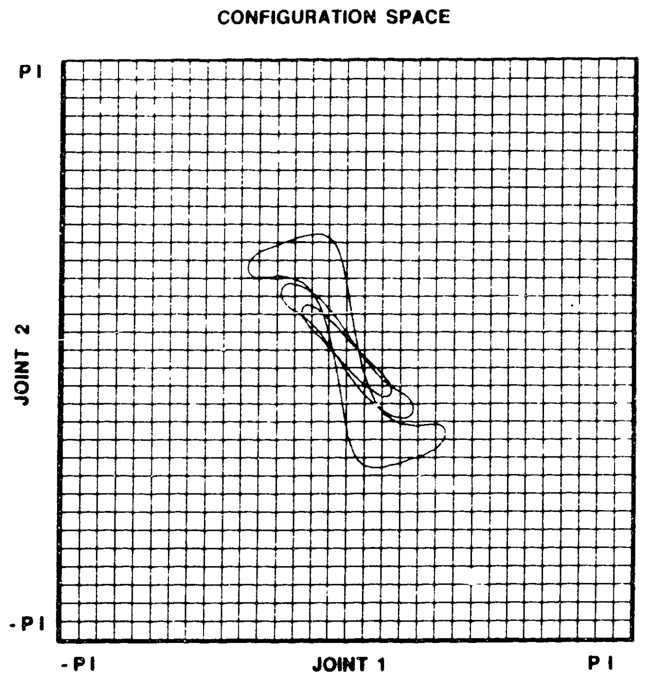

Figure 11: 2" Obstacles on x-axis; C-Space

In order to approximate the radial dependence of obstacle placement on configuration space images, the three types of obstacles are considered separately. The chosen approximation for link2-front obstacles is illustrated in figure 12. In figure 12, 8 polygonal approximations of the true configuration space images for 8 placements of the obstacle along the x-axis are shown superimposed. In all, 14 empirical tracings were obtained, and 14 polygonal approximations were created. Each polygon has 12 vertices and odd symmetry. Thus, an approximating polygon can be fully described by an ordered list of 6 points corresponding to six sequential vertices. The entire database for link2-front obstacles consists of 84 points (coordinate pairs) in configuration space.

The relatively coarse spacing of samples along the x-axis would be inadequate if the polygonal approximations were used identically to define configuration space areas corresponding to obstacle locations between sample points. A sparse database is made possible by interpolating images between sample points. Interpolation of images is performed by linear interpolation of vertex locations

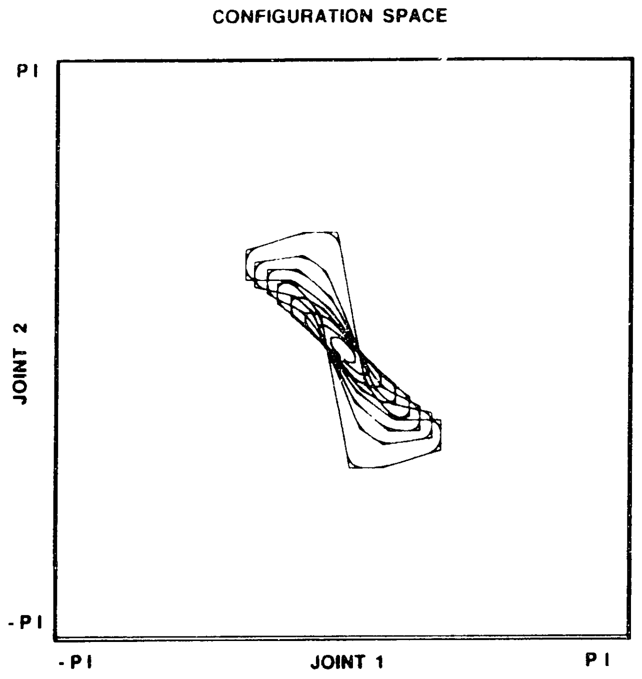

Figure 12: Polygonal Obstacle Approximations; Link2-front

from the exact locations of corresponding vertices in the database. This is made possible by ordering the vertex coordinates in the database in a consistent manner such that there is a logical correspondence among vertices of various sample images. In specific, each of the polygons in figure 12 has a vertex which is uniquely identifiable as the "upper left corner" of each polygon. This vertex has been given the label "0" for each stored polygon. Vertices "1" through "5" are identified as the vertices encountered sequentially while tracing the contour of the polynomial counterclockwise, starting at vertex "0"

An example of linear interpolation of vertices follows. The coordinates of vertex "0" are (θ_1, θ_2) = (-2600,700) (in encoder units of 0.011 deg/unit) for the link2-front polygonal approximation of an obstacle at radius 42 cm. Vertex "0" has coordinates (-2800,800) for the polygon corresponding to an obstacle at radius = 38 cm. For an obstacle placed at radius = 40 cm, an equivalent vertex 0 may be computed by linear interpolation to be at the point (-2700,750) in joint space.

The complete process of computing a link2-front obstacle consists of: 1) convert the obstacle's center coordinates in (x, y) to polar coordinates, (r, α); 2) find the least upper bound and greatest lower bound radii of stored obstacle data with respect to r; 3) compute the coordinates of each vertex of the approximating polygon from interpolation of the corresponding vertex locations of the upper and lower bound images in the database; generate remaining vertices from odd symmetry; 4) increment each computed coordinate by an amount equal to the polar angle of the obstacle center in task space; 5) join successive vertices with vectors in joint space. The resulting contour encloses a region in joint space which is a close approximation of the set of robot poses which would result in a collision between some part of the circular obstacle and some part of the front of link 2.

An identical procedure is followed for constructing link2-rear obstacles. Twelve sided polygons are also used in the link2-rear database. Ten samples were stored, covering the range from 12 cm to 38 cm, outside of which all points within the obstacle set lie outside regions III and IV of figure 7. The database for link2-rear obstacles contributes another 60 points of required storage.

Link1 obstacles are also stored as a function of radius, measured at discrete radii. A single, one-dimensional point is all that must be stored to describe a link1 obstacle. For a circular obstacle on the x-axis, the link1 configuration space contribution appears as a vertical stripe. The width of the stripe is an adequate description of the forbidden region, which is centered about the origin in configuration space. For points off the x axis, the center of the stripe is shifted by an amount equal to the polar angle of the obstacle center position. Linear interpolation of stripe widths is performed between bounding samples stored at selected radii. Thirteen data points were stored at selected radii within regions II and III of figure 7.

Link2-front and link2-rear obstacles become difficult to measure when the physical obstacle is within reach of link1. The presence of link1 prevents a complete trace of the contours for link 2 interference. However, the existence of link1

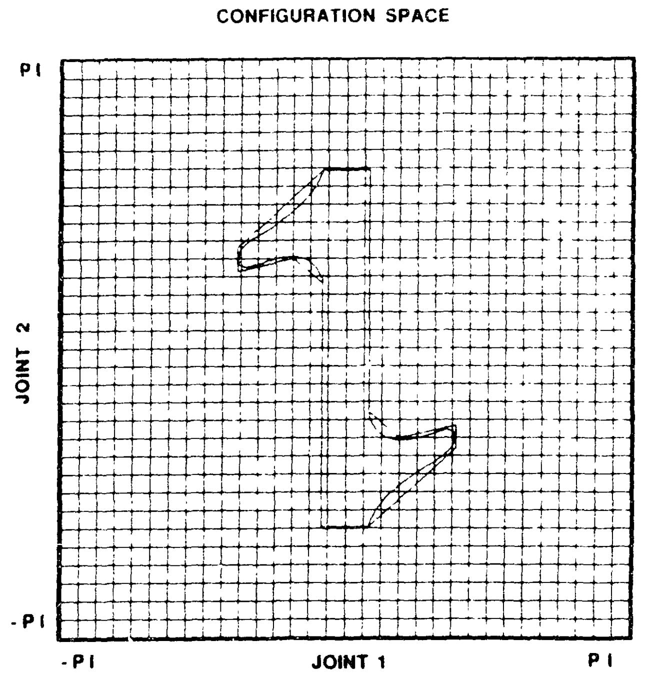

Figure 13: Approximation of Link2-front Obstacle in C-Space

obstacle stripes on configuration space makes a detailed description of link2 obstacles pointless within the regions occluded by the link1 stripe. Thus, the bounding polygon approximations may make liberal assumptions about the link2 obstacle shapes within the regions of link1 obstacles. An example is shown in figure 13. Only the curved portions of the graph correspond to actual contact between the front of link2 and the obstacle on the x-axis. The vertical line segments of the graph correspond to the limits of link 1 travel with respect to the obstacle. The true shape of the link2-front obstacle within these bounds has not been determined. The approximating polynomial simplifies the obstacle shape within these bounds by employing horizontal and vertical line segments. No loss of accuracy is introduced by this model, since the unknown region of the link2-front obstacle is covered by the link1 obstacle.

The database of polygonal approximations was constructed for a density of sample points which was empirically judged to provide good accuracy for interpolated polygons. Given that linear interpolation between stored samples is valid, a simple means for transforming task space "swept" obstacles into configuration space can be described.

First, consider an obstacle which has been swept tangentially (at constant radius) in task space. The joint-space images of the obstacle corresponding to the start and finish locations of the sweep are identical, except for a diagonal shift in joint space. Consider the approximating polygons for the start and finish locations. Connect each of the corresponding vertices with line segments (vertex "0" to vertex "0" etc). The resulting graph forms a closed region (containing a number of superfluous internal line segments) which completely encloses the forbidden states corresponding to the entire swept obstacle region. If the approximating images at the two endpoints were perfect, the image "swept" in configuration space by joining corresponding vertices would perfectly describe the exact transformation of the swept region in task space. This is a consequence of the fact that the transformation can be expressed as a composite function: $f_c(\alpha, r) = f_\alpha(\alpha, f_r(r))$. Since the transformation in polar angle is linear, linear interpolation based on this variable is exact.

Task-space obstacles which are swept radially do not map as easily into configuration space. However, they may be approximated in configuration space using incremental linear interpolation. The database of configuration space images was constructed such that linear interpolation between successive samples was deemed sufficiently accurate. To the same accuracy, an obstacle which is swept between successive stored radial locations may be approximated in configuration space by joining corresponding vertices of the stored polygons. A long radial sweep in task space may be realized in configuration space by multiple piece-wise linear interpolations (linear vertex connections) among successive stored mappings.

A swept motion which is both radial and tangential in task space is swept in configuration space in the same manner. Intermediate positions of the swept obstacle are computed, and respective vertices are connected. The choice of density of intermediate positions is based solely on the change in radius in task space, since linear interpolation of the tangential variations is exact over arbitrarily large tangential sweeps.

Using swept obstacles, task-space features such as lines and arbitrary contours can be mapped efficiently into configuration space.

§6. Discretization of Configuration Space

The process detailed above describes the construction of polygons in configuration space which enclose forbidden regions. To detect if a robot is near an obstacle, it would be necessary to select the point in configuration space which corresponds to the robot's position, and determine the distance to each line segment in the configuration space map. To avoid collisions with obstacles, it is necessary to prevent the robot from crossing any line segment in the map. The process of determining the distance to each line segment can take arbitrarily long as the number of obstacles grows. In order to make the representation more efficient, the configuration space has been discretized.

To discretize configuration space into "pixels" a pixel is turned "on" if it contains any part of any line segment of configuration space bounding polygons. (A slightly more conservative discretization has actually been implemented). Figure 14 shows the boundaries of five circular obstacles transformed into configuration space and discretized into a space of 128x128 pixels. The original bounding polynomials are inscribed in two of the images. In this example, 120 line segments are present before discretization. To determine if a robot is about to collide with an obstacle, it would be necessary to compute the distance to each of the 120 line segments. Once discretized, though, a robot can tell if it is in danger by examining whether the pixel which it occupies is "on"

Two additional features of discretized configuration space have been implemented which aid in the computation of dynamic obstacles and in the realization of servo-level obstacle avoidance. That is, each pixel has "gray" levels, and four

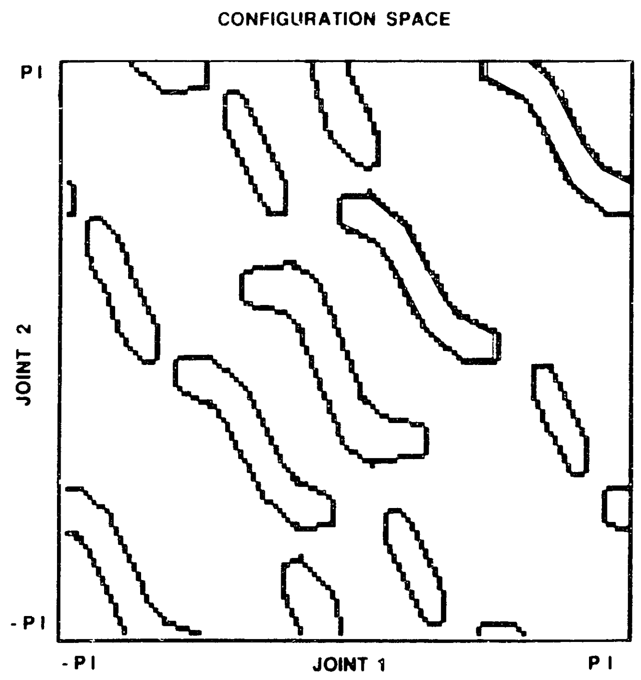

Figure 14: Obstacles in Discretized C-Space

sub-categories. The four sub-categories indicate edge-normal (surface-normal) information. The gray level indicates the number of contributors to each "on" pixel.

For automatic obstacle avoidance, it is important to know not only that a potential collision is near, but also which direction to move the robot to avoid the possible collision. The direction which the robot should take to avoid collision is the direction of the outward surface normal of the configuration space obstacle. Once a configuration space obstacle has been discretized, each of its elements has a surface normal which points in one (or more) of four directions. The interpretation of surface normals within a pixel is as follows. If a robot pose (or point automaton) is within an "on" pixel, then by moving in the direction of any of the surface normal components (i.e. up, down, left or right) the robot will not enter the interior of the forbidden region bounded by the "on" pixel. Alternatively, an "on" pixel can be thought of as repelling the robot in the direction of its edge (surface) normal. For each computed obstacle, edge normal information is implicit in the ordering of the vertices. Since the vertices have been recorded in counterclockwise sequence, each edge component is actually a vector. Rotation of each edge vector by -90 degrees provides the corresponding edge normal vector. The signs of the θ_1 and θ_2 components of the edge normal vector determine the directions of repulsion for each "on" pixel derived from a given edge. The utility of this scheme will become apparent in the next chapter.

The use of gray levels for each pixel enables extensibility to moving obstacles. When an obstacle moves, its configuration space mapping can change dramatically. The process of computing edges of bounding polynomials is efficient enough to keep track of rapidly moving obstacles. However, once an obstacle has been discretized into pixels, the original obstacle loses its identity. The new representation fully describes the effect of continuous obstacle mappings, but, once discretized, it is impossible to tell which physical obstacle was responsible for affecting any given pixel. Thus, as an obstacle moves, it is hard to decide which pixels should be restored to their "off" state.

A natural approach to resetting pixels when an obstacle moves is to construct an entire new map from a reset state (all pixels off). For a single obstacle, this approach would be adequate. However, complex environments can involve a combination of static obstacles and dynamic obstacles, the former being more common. Recomputing the discretized configuration space map would require recomputing all of the obstacles present; not just the obstacle which moved.

An alternative candidate approach is to keep track of the old position of a moving obstacle, erase (turn off) all pixels corresponding to its former position, and turn on all pixels corresponding to its new position. Pixels to be erased can be computed by retracing the old image, or by storing a list of active pixels associated with each obstacle. Storing a list of pixels, though, would quickly consume exorbitant memory as the number of obstacles increased. In either event, there is a problem with erasing pixels of an old image, since multiple obstacles may affect the same pixel. That is, the joint angles corresponding to a single pixel in configuration space may correspond to multiple intersections between the robot's envelope and the volumes of various obstacles. Obstacles which appear widely separated in task space may have configuration space images which are largely overlapping. Thus, erasing a pixel which appeared in the mapping range of the old position of an obstacle can result in inadvertently allowing robot motion in a region which results in collisions with a different obstacle.

A solution to this problem utilizes gray levels. As each obstacle is discretized all affected pixels on the border of the obstacle are incremented; i.e., each pixel holds a count of the number of contributors to its "on" state. As an obstacle is removed, each pixel affected by the obstacle's former state is decremented. When the gray level of a pixel is reduced to zero, then the contained region of configuration space is guaranteed to be free space. Static obstacles and dynamic obstacles may share this same representation. Further, multiple processors may operate on the same map without knowledge of each other, and the interpretation of "on" and "off" pixels will remain valid.

In the algorithm implemented, 2-dimensional configuration space has been discretized into 16k regions (128x128 pixels), each with 128 "gray" levels (2^8, or 1 byte of storage per pixel). Obstacle centroids are continuously updated by the sensing system, and configuration space maps are continuously updated. In the process, a "new" obstacle transformation is added to the map by incrementing the value of each pixel which contains some part of the polygon's boundary.

The "old" obstacle state is stored in terms of the former position of the centroid in task space. After computing the new obstacle image, the former image may be erased from the map by recomputing the configuration space image from the old task space location, and decrementing all affected pixels. The order is important. New images are overlayed before erasing old images so that obstacles do not seem to "disappear" or open up voids during computations. The configuration space map may thus be accessed at any time, not just at the completion of a transformation. A best approximation of the state of the world is always available to the robot.

§7. Computation Efficiency

To this point, the proposed configuration space transformation technique has been demonstrated to have negligible memory requirements for the image database, and generality of shapes based on circular primitives which may be superimposed and swept. The practicality of the technique hinges on the speed at which the primitives can be converted from task space to configuration space.

The interpolated polygonal approximation method was implemented and evaluated for 2" diameter circular obstacles. The computer used was an MC68020 based single board computer running at 16.7 MHz. Configuration space maps were stored in off-board RAM which was accessed via a VME bus backplane. The code was written in "C"

The polygonal image approximation database consisted of 144 points in 2-D space and 13 points in 1-D space. Interpolated polygons corresponding to the image space of arbitrarily placed 2" obstacles were computed extremely fast. Polygon constructions were timed at 1.67 ms (600 Hz) for obstacles in region II of figure 7, 3.33 ms (300 Hz) in regions III and IV, and 1.67 ms in region V. Roughly, it takes 100 microseconds to interpolate a vertex. The algorithm would behave identically for point obstacles, and circular obstacles of arbitrary radius.

The process of discretizing the polygonal obstacles consumed a considerably longer time. Each obstacle is traced about its perimeter to determine which pixels in configuration space are affected, and each of these is incremented. An obstacle update also requires an identical computation to erase its former image. Complete representative cycles of update plus erasure were timed with the following results: Obstacles in region II took 20 ms; obstacles in region III required from 20 ms to 30 ms; obstacles in region IV took 14 ms, and region V obstacles took 7 ms. Thus, the discretized configuration space map can be updated at nominally 50 Hz, 33

Hz in the worst case, for a single moving circular obstacle.

In implementation, an obstacle update is performed only if the obstacle has moved since the last time its transform was computed. With this addition, the complexity of a dynamic environment depends on the number of obstacles which move simultaneously, and not on the number of movable obstacles.

The processing speeds noted above are for a single processor. As noted earlier, the use of discretized configuration space with gray levels supports the use of multiple processors. A naive application of multiple processors would be to assign one or more obstacles to each CPU. The individual processors would be entirely compatible without requiring any knowledge of each other. The assignment of obstacles to processors, though, would not generally balance the computation load evenly. For example, all of the obstacles assigned to one processor might be standing still while all of the obstacles assigned to another processor are moving. In this case, the additional processor does not improve the speed of the configuration space map evolution.

Alternatively, a circular list could be maintained of all obstacle positions and their former positions at the time of their most recent transformation. A semaphore would indicate the current obstacle to be transformed. As a CPU becomes available, it would select the flagged obstacle for transformation, note the current and former positions, update the record of most recent position of transformation to be the current obstacle position, advance the semaphore to the next obstacle in the cycle, and begin its task of transforming the selected obstacle. If the "new" obstacle position is the same as the "old" obstacle position at which the obstacle was previously transformed, a new transformation will be skipped. The CPU will quickly return for a new assignment. In this manner, the computational load of obstacle transformations will be evenly balanced among N processors. Thus, the configuration space transformation rate will be N times faster with N processors (within bus bandwidth limitations). Each processor would run the exact same program and operate on the same map in global memory. The system is robust in that the failure of any one processor will not invalidate the transformation process; it would only affect the speed. In fact, as long as at least one processor continues to function, the configuration space transforms will still be rigorous, albeit slower.

The rate of transformation computations for obstacles in 2-D is sufficiently fast. Speed of transformations alone, however, is not sufficient to safely monitor high-speed moving obstacles. A fast-moving obstacle should be represented in configuration space not only by its current position, but by the swept positions along its anticipated trajectory for some critical time in the future. The critical time is a function of both the transformation rate and the speed and proximity of the robot. Discontinuities are of particular concern. Near a discontinuity of the configuration space map, e.g. near the robot's elbow, a small motion of an obstacle can correspond to a major alteration of configuration space. The moving obstacles should be represented by their "blurred" images of along their anticipated respective trajectories for a time Δt in the future. The blurred view in task space is conceptually the same as a photograph of the moving obstacles taken with an exposure time of Δt. Uncertainty in the anticipated trajectory of an obstacle is reflected in additional blurring of the obstacle to cover the region of uncertainty.

§8. Generalization to Higher Dimensions

The transformation technique which has been implemented in 2-D is generalizable to higher dimensions, albeit with potentially severe penalties in computation and storage requirements. As discussed earlier, though, 3-D configuration spaces are arguably adequate for most industrial robots. Extensions to 3-D are considered specifically.

Previous deductions of set properties for configuration space transformations were not restricted to 2-D. Constructive geometry via point transformations applies in higher dimensions. In 3-D, points and spheres are useful task-space geometric primitives. Constructing unions and swept volumes are useful operations in the efficient transformation of arbitrary shapes in higher dimensions. Further, map discontinuities also occur in higher dimensions. Discontinuities occur at boundaries at which an obstacle "type" appears or disappears (e.g., link "i" obstacles). Within regions of continuity, interpolation of configuration space volumes is valid.

The chosen polygonal approximation of 2-D configuration space images generalizes naturally to polyhedra in 3-D. Sample forbidden volumes in configuration space can be enclosed by a shell, (not necessarily convex), consisting of facets. Each facet may be described by its bounding vertices. The order of the vertices is sufficient to determine the direction of the surface normal of the facet. Thus, volumes in configuration space may be approximated by polyhedra and stored compactly in terms of the coordinates of the vertices. Memory requirements for this approach are modest.

The number of polyhedral volume samples should be kept to a minimum. The savings incurred by the use of an intermediate coordinate system in 2-D (polar coordinates) is also possible in 3-D, depending on the robot involved. For any 3 d.o.f. robot, if two consecutive joints are revolute and their axes intersect orthogonally, an intermediate frame of spherical coordinates centered about the intersection point should be used. In this intermediate frame, the configuration space mapping of a point is geometrically identical for all points at the same radius. Variations in azimuth and altitude correspond to pure translations of obstacle shapes in joint space. Thus, sample volumes (polyhedra) need to be stored as a function of radius only.

Virtually all industrial robots which are not Cartesian or SCARA satisfy the criterion for simplification. This is no coincidence, since the same criterion simplifies the inverse kinematics of the robot. Example robots which satisfy the criterion for the spherical coordinate simplification include the Stanford Arm, the Unimation Puma series, and the M.I.T. Direct-Drive Arm. Industrial robots with successive rotational joints are built either with the joint axes parallel (as in SCARA designs) or orthogonal. Joints with skewed axes are certainly possible, but are never built in practice. With rare exceptions, orthogonal axes of rotation are designed to intersect.

The remaining two popular classes of robots, SCARA and Cartesian, also have simplifications for their configuration space transforms. Configuration space transforms are especially simple for Cartesian robots, since configuration space is similar to the actual workspace. For SCARA robots, an intermediate frame of cylindrical coordinates is useful. Points of equal radius and altitude have identical geometries in configuration space. Variations as a function of altitude are linear, at least within specified regions. Points within the axial and radial reach of links 1 and 2 must be considered separately. The swept volume of links 1 and 2 defines a region in which additional obstacle "types" appear (corresponding to collisions with link1 or link2). The transformation discontinuity across the boundary of this region may be treated in the same manner as described in 2-D. The configuration space transformations of each obstacle "type" are still linear in z and α, requiring only a one-dimensional database in r.

Discretization of 3-D configuration space can be performed analogous to that in 2-D. For each polyhedron, each face is swept to determine which "voxels" in 3-D contain some part of the face. Each affected voxel is incremented in one or more of its directional categories. For a cube in 3-D, there are six possible surface normal components, as opposed to the four directions in 2-D. The memory required for a map in 3-D with the same resolution as the 2-D example is: 128x128x128x6 bytes, which is 12Mb of memory. This memory requirement is significant by current standards, but is not unreasonable. Storage of a map of this size is possible on the computer system used in the experimental setup described in chapter 2. It may be expected that this magnitude of memory will be considered minor in the near future.

A more serious question is the computational penalty involved in higher dimensions. The computational speed of the approach described here depends on the number of affected voxels. As an illustrative example, consider a square in 2-D configuration space with dimensions of 10 pixel widths. If the square is aligned with the coordinate axes, then the number of pixels affected by an edge is 10. The number of pixels affected by tracing the perimeter of the square would be 40 (a primitive algorithm would double count corners). This example may be compared to that of a cube in 3-D configuration space aligned with the axes of the coordinate system. Let the face of the cube have the same dimensions as the square considered in 2-D. The number of voxels affected by a face is 100, and the number of voxels affected by all six faces is 600. For this example, the configuration space discretization of the faces of the cube in 3-D may be expected to be 15 times slower than the discretization of the edges of the square in 2-D. The comparison between computations in 3-D vs 2-D depends on the specific images to be discretized. Very roughly, though, 3-D computations may be expected to take 1 to 2 orders of magnitude longer than the 2-D computations. Since 2-D transformations are quite fast, practical implementation of 3-D transformations of moving obstacles should be possible using parallel processing.

§9. Conclusions

In this chapter, a method has been presented for computing and storing configuration space representations of obstacles. The method depends on two important properties of configuration space maps: 1) the transforms of a union of obstacles is the union of the transforms of the obstacles; and 2) the transforms are continuous within known regions. The first property justifies the use of simple modeling primitives (points, circles, spheres) in the construction of more complex shapes (via union or swept region of points, circles, spheres). Property (1) also justifies reducing the transformation of a task space region (volume) to the transformation of is border (surface). Property (2) justifies the use of interpolation among discrete "values" (images) of the configuration space transformation. The use of an intermediate coordinate frame in task space (polar coordinates) reduces the problem to a one-dimensional functional interpolation.

The use of discretized configuration space permits rapid interpretation of obstacle transformations. Further, the use of gray levels in discretized configuration space makes the computation of incremental evolution of the configuration space obstacles possible. In addition, it enables the efficient use of parallel computing.

The proposed method was implemented in 2-D for the rapid transformation of circular obstacles. Transformations were performed (nominally) in 20 ms per obstacle per CPU.

Extensions of the implementation to 3-D were considered and found to be practical or nearly practical with existing technology.

CHAPTER V

OBSTACLE AVOIDANCE USING REFLEX CONTROL

§1. Introduction

In Chapter 3, a robust technique for high-speed robot control was presented. In the treatment there, it was implicitly assumed that the motion resulting from the controller was feasible. If obstacles are present within reach of the robot, then possible collisions must be prevented, especially when the robot has high-speed capability. In Chapter 4 it was shown how obstacles may be transformed into forbidden regions in configuration space. Collisions with obstacles are avoided provided the trajectory resulting from the control scheme does not enter any of the forbidden regions. The controller, however, does not use any explicit trajectory pre-computation, so path intersections with forbidden regions are not known a priori. In this chapter, a technique called "Reflex Control" is presented which reconciles high-speed robot control with guaranteed obstacle avoidance. Reflex control derives its name from attributes of its behavior which are analogous to reflexes: it is normally transparent to planned activities, but is capable of rapidly responding to environmental conditions by assuming motor control with precedence over higher levels of reasoning.

Reflex control is derived from a variation on potential function control. In the following, potential functions are used to describe a compatible formalism for combining optimal control with guaranteed obstacle avoidance. First, potential functions are considered in one dimension. Several important concepts are introduced, including logical combinations of potential functions, the definition of an "arrest point", the computation of a limited lookahead window, and the determination of reflex subgoals. The concepts are then generalized to higher dimensions. Finally, implementation results in 2-D are presented.

§2. Potential Function Control: Background

In potential function based control, scalar functions are defined over the space spanned by a system (e.g., the joint space or the workspace of a robot) and are used to define control efforts. Individual potential functions are combined, and a virtual force field is computed by taking the gradient of the net potential. The system is made to imitate a response to the virtual forces by exerting actuator efforts which are equivalent to the effect of the virtual forces acting on the system.

The use of potential functions for manipulator control has been described by various researchers. Khatib [20] and Myers [33] invoke the use of potential functions in "operational space" to control a robot for which the dynamics are decoupled through a transformation in feedback. In this case, potential functions for attraction, repulsion and damping are all expressed in task (Cartesian) coordinates; only robot joint limitations are expressed as potential functions in joint space. Goals exert virtual attractive forces on the robot, and obstacles exert virtual repulsive forces. Multiple obstacles are protected by individual repulsive potential fields which are combined by arithmetic superposition. The sum of all virtual forces is used to describe an apparent net force on the robot. The robot is controlled to imitate Cartesian dynamics in response to the virtual forces. Khatib defines dynamic constraints of maximum acceleration and maximum velocity in task coordinates.

Takegaki and Arimoto [48] describe the effects of potential functions in terms of a Hamiltonian formulation. They show that potential function control in joint space using joint attractors and joint damping functions is globally stable, provided there are no local equilibria. The proof for stability using potential functions in task space depends on the existence of a uniquely invertible Jacobian, i.e. avoiding singularities and ignoring left hand/ right hand type multiple solutions. They go on to show how potential function control is compatible with performing kinematically constrained tasks, e.g. turning a crank.

Andrews and Hogan [1] presented potential function control in the context of impedance control. A potential function approach was used to avoid a moving obstacle while heading for a fixed goal. In addition, the same approach was shown to be effective in turning a crank, where the path constraint of the crank handle was not specified a priori.

Krogh [24] introduced enhancements on potential function control through the use of a "reserve avoidance time." Here, potential functions increase their amplitudes depending on the minimum braking distance to stop a manipulator from colliding with an obstacle. Krogh also proposed the nonlinear, (and non-conservative), operation of setting an obstacle's potential function to zero if the manipulator is headed away from the obstacle. He also described combining optimal control with obstacle avoidance by adding the equivalent force vectors of optimal control and of each obstacle potential field.

Koditschek [23] applied topological analysis to potential function based control schemes to derive a procedure for eliminating local minima from a control scheme. In this work, the potential functions are added linearly, and grown individually in a manner which assures at most one stable point.

In prior work, potential functions have been used to describe the presence of obstacles in terms of their direct influence on a robot's motion. Such a description permits inherent obstacle avoidance at low levels of a control hierarchy, as opposed to conventional approaches in which a robot is controlled to track a pre-computed trajectory. By incorporating obstacle avoidance directly into motor control levels, a robot may respond interactively in dynamic environments. Interactive control is desirable, since it enables the robot to instantly respond to a changing environment. In each of the preceding works, obstacles are theoretically protected by making the respective protective potential functions wide and/or steep enough. For the proofs of obstacle avoidance to remain valid, however, it was necessary in each case to assume that the robot was capable of infinite acceleration. In addition, the protective fields described are typically overly conservative, resulting in an unnecessarily large influence on the robot. Overly influential potential functions often lead to obstruction of the goal location or obstruction of a valid path to the goal. In general, the proofs of guaranteed safety remain valid in cases where the robot is moving at speeds well below its capacity. Enough extra actuator effort is then available to exert the equivalent efforts prescribed by the potential fields.

In the present work, begun in [35], dynamically decoupled systems are considered. Potential functions are used as a basis for reconciling optimal control with guaranteed obstacle avoidance. Explicit consideration of manipulator dynamics and actuator limits is incorporated in the potential function formulation using an energy interpretation. An absolute minimum potential function influence is derived which is guaranteed both safe and feasible, at least for dynamically decoupled robots.

§3. Potential Function Control in One Dimension

In this section a single link of a frictionless, dynamically decoupled manipulator is considered. The dynamics of the robot link are assumed to be described by:

$$\begin{aligned} I\ddot{\theta} &= u \\ \text{where} \quad |u| &\leq u_{max} \end{aligned} \tag{V.1}$$

The preceding equation is a good model for each of the links of the robot described in Chapter 2. Since the links are dynamically decoupled, their controls may be considered separately.

The rotational kinetic energy, T, of a manipulator link with angular velocity

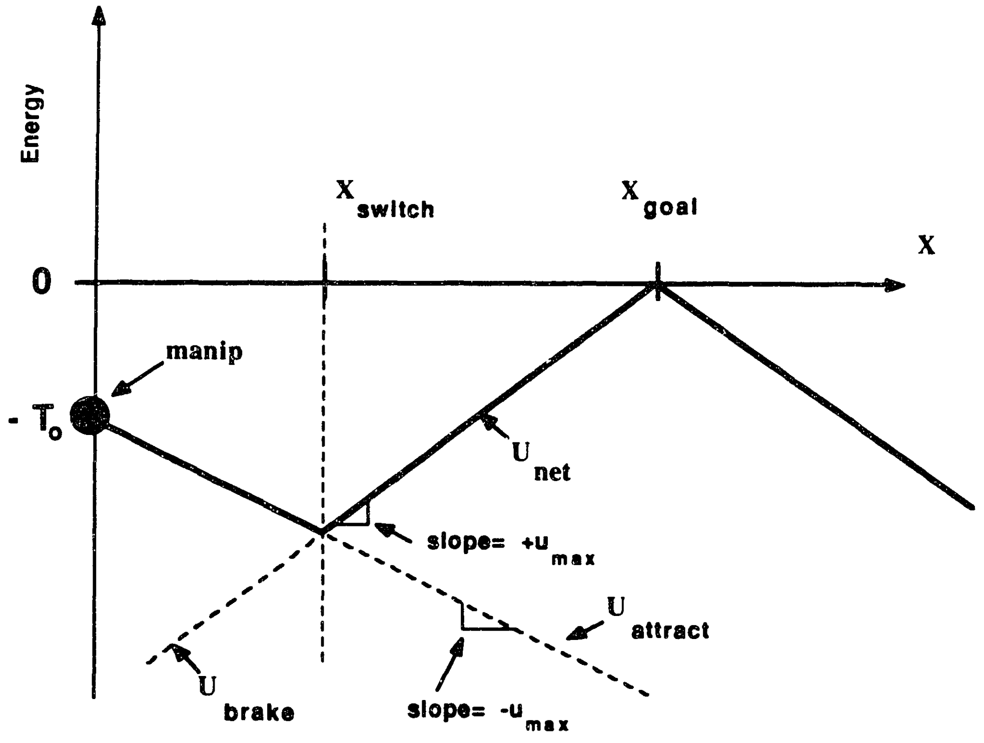

Figure 1: 1-D Potential Function Control of a Pure Inertia

ω is:

$$T = I\omega^2/2 \tag{V.2}$$

Attraction to a target position, θ_{goal}, may be achieved by imposing a potential function sloped toward the goal. In figure 1, such a function is shown, labeled $U_{attract}$. In this figure, the manipulator is defined to start at $\theta = 0$ with initial kinetic energy T_0. The attractive potential function is defined to have a potential energy of $-T_0$ at the manipulator's starting location. Under the influence of the attractive well, the manipulator will accelerate toward the goal, gaining additional kinetic energy exactly equal to the potential energy decrease defined by the attractive function.

Acceleration towards the goal, however, is not sufficient in itself to obtain desirable manipulator control. Attraction alone would result in oscillation about the goal position. Deceleration is required to absorb all of the manipulator's kinetic energy to bring it to rest at the goal. Any feasible potential function with an energy peak of $U_{max} = 0$ centered on the goal position will perform the desired braking. Such a braking potential is shown in figure 1, labeled U_{brake}. The two potentials, attraction and braking, are combined into an appropriate control policy through a logical (vs arithmetic) operation. For a set of potential functions $\mathbf{U}$ containing individual potential functions $U_i(\theta)$, the effective potential U_{net} at any location θ is given by:

$$U_{net}(\theta) = U_j(\theta) \text{ where } \forall U_i \in \mathbf{U} : \;\; U_i(\theta) \leq U_j(\theta) \tag{V.3}$$

That is, the highest energy potential function at each point defines the net field at that point.

Actuator efforts are exerted on the manipulator proportional to the gradient of the net potential function. In one dimension:

$$u = -\frac{d}{d\theta} U_{net}(\theta) \tag{V.4}$$

If $U_{attract}$ decreases monotonically towards the goal, and if U_{brake} increases monotonically towards the goal, then the logical combination of potential functions given by equation V.3 results in braking within some neighborhood about the goal and attraction outside that region. The transition from attraction to braking occurs at the point labeled $\theta = \theta_{switch}$ in figure 1. Besides monotonicity, no specification of the potential function shapes has yet been prescribed. Any choice of attractive valley and repulsive hill will result in driving the manipulator to rest at the goal position without overshoot, provided the potential energy zero-reference and peak are selected as described, and provided the initial position of the manipulator does not lie within the region $\theta < |\theta_{goal} - \theta_{switch}|$. If the manipulator does start within the braking region, then the initial energy is too high to prevent overshoot within the saturation limits of the actuators.

The shape of the potential functions determines the magnitude of forces exerted on the manipulator, as given by equation V.4. For the potential functions to prescribe feasible forces, the slopes of the functions must not exceed the effort saturation bounds of the actuator. For the actuator limits described in equation V.1, the slope of the potential functions should not exceed u_{max} in magnitude. Since any set of potential functions with the defined maximum and zero reference will bring the manipulator to the goal without overshoot, it is clear that potential functions with maximum allowable slopes will perform the task in minimum time.

In figure 1, both the attractive and braking functions are constructed with maximum allowable slopes. As a result, the manipulator would be driven by bang-bang control, with a switching point at θ_{switch}, and would come to rest at the origin in minimum time. Thus, for this simple system, optimal control policies are easily described with a logical combination of simple potential functions.

Arguments of the preceding discussion can be applied to obstacle avoidance as well. In figure 1, if the goal position is interpreted instead to be the boundary of an obstacle, (as represented in joint space), then the proposed braking function forces the manipulator to come to rest at the obstacle boundary, thus preventing collision. The attractive part of the net potential field may be an arbitrary subset of the control fields constructed for goal interception. Regardless of the magnitude or shape of this field, any obstacle potential function with energy 0 at the obstacle's (joint space) boundary will protect the obstacle from collision with the manipulator.

Additional aspects of logical field combinations are illustrated in figure 2. In Case A, an obstacle protection function, U_{obs}, is constructed over a large obstacle centered at θ_{obs}. The energy of the function at the obstacle boundaries is 0, and the slope of the function corresponds to the maximum feasible actuator effort. Part of a target attraction function, U_{goal}, is also shown. The fields are combined using equation V.3.

In Case B, the obstacle is represented as two smaller abutting obstacles. Two protection functions are constructed, each with energy 0 at their respective obstacle boundaries. When the obstacle functions and the original goal potential

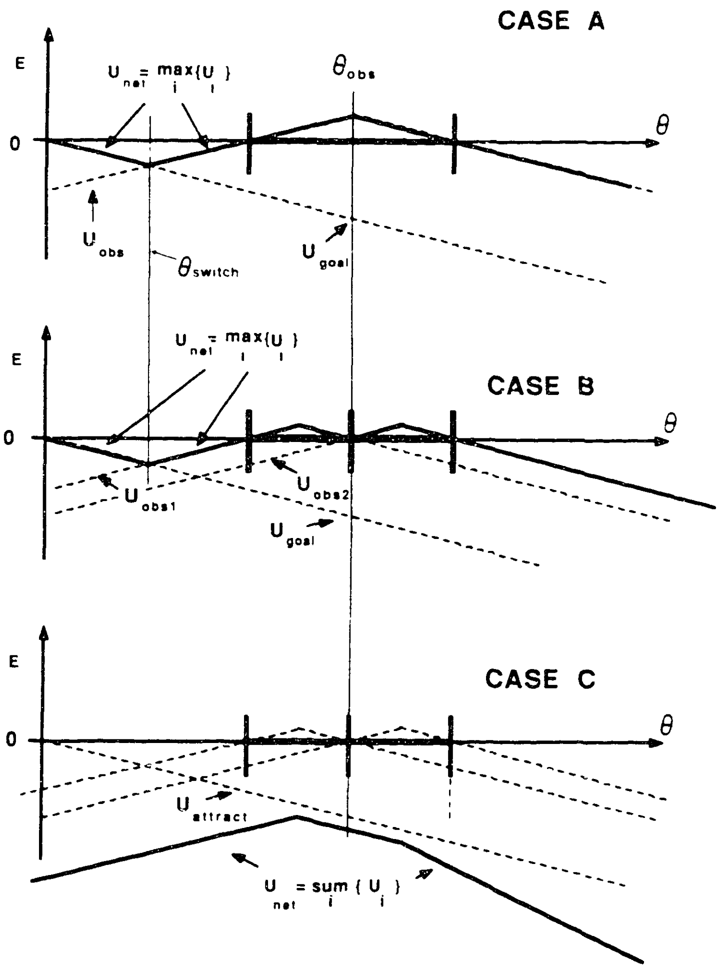

Figure 2: Logical and Arithmetic Potential Function Combinations

function are combined logically, the resulting field has the same effect as Case A, as it should. The switch point for transitions from manipulator acceleration to manipulator braking occurs at precisely the same location, all slopes define feasible actuator efforts, the obstacles are guaranteed safe, and they exert minimum influence on the original goal control policy.

In Case C, the situation is identical to Case B, except the potential fields are superimposed arithmetically, as proposed by previous researchers. The resulting potential field has lost all of the desired properties. The control efforts prescribed by the net potential field include magnitudes which exceed the feasible limits. The original goal attractive field is completely swamped by the obstacle fields, which push the manipulator out to infinity. Further, the shape of the net potential no longer coincides with the actual obstacle boundaries, which, in some cases, will leave the obstacle borders unprotected. If the long obstacle in Case A is subdivided into many small obstacles, potential combination via equation V.3 will still give the same control result as Case A. If the many individual obstacle potentials are summed, however, the result will be correspondingly worse than the two-obstacle example of Case C.

Logical combination of potential functions provides a net potential field which conserves energy. Another useful logical operation on potential fields is the suppression of any repulsive field from which the manipulator is receding. Heuristically, it is apparent that a field which is imposed to prevent a collision is unnecessary if the manipulator is heading away from the obstacle. While this operation is useful and intuitively appealing, it is non-conservative. For example, if the manipulator climbs up a potential hill, comes to rest, then begins to slide back down the potential hill, when it returns to the bottom of the hill it will have the same energy with which it started. If, however, the potential is suppressed when the manipulator reverses direction, energy will not be conserved; the manipulator will not regain the kinetic energy which was converted to potential energy by the obstacle field. Selectively suppressing potential fields is nonetheless a useful operation; a new energy computation of the remaining fields must be performed, however, if an active potential field gets suppressed.

Energy conservation is also violated if the potentials are moving. In the static case, obstacle protection is guaranteed by setting each obstacle potential to zero at its respective obstacle boundary. With moving potentials, however, energy is not conserved, and obstacle protection is no longer guaranteed. This is not merely a flaw of potential-based control schemes, though. In fact, no control law whatsoever can guarantee prevention of collisions in a space of moving obstacles. Consider, for example, the one dimensional case of a manipulator situated between two obstacles which are both moving towards the manipulator. Collision with the obstacles is inevitable, regardless of the control policy. This example represents a geometric limitation which applies as well in higher dimensions.

Analogous cases in higher dimensions can be easily visualized in terms of configuration space, in which the obstacles are represented in terms of forbidden sets of robot joint angles. As shown in chapter IV, configuration space obstacles can easily overlap, although they might be widely separated in task space. Even a seemingly sparse field of obstacles can result in configuration space forbidden regions which combine to form a closed hull surrounding the manipulator's current configuration point. In such an event, it is geometrically impossible for the manipulator to escape from the obstacle confines. If the obstacles are moving, then the surrounding hull of obstacles in configuration space may collapse inwards on the manipulator, resulting in inevitable collision. In this scenario, the impossibility of escape is certain, although the obstacles may move arbitrarily slowly and the robot may be arbitrarily agile. The restriction is purely geometric, and no control policy whatsoever can guarantee safety.

Under potential function control, when the controller can not guarantee obstacle avoidance, the situation will be flagged by a violation of conservation of energy. Violation of conservation of energy does not imply that a collision is certain, however. Rather, it indicates that the safety guarantee based on energy conservation does not apply. Some types of nonconservative operations may preserve safety guarantees although energy is not conserved. For example, the nonconservative operation of suppressing a potential field when the manipulator is receding from the respective obstacle does not conserve energy, but the effect is purely dissipative. For static obstacles, invoking this operation will not cause collisions.

§4. Optimal Control With Obstacle Avoidance in One Dimension

In the preceding section, it was shown how the construction of potential functions can be used to describe time-optimal control (in the absence of obstacles) and guarantee obstacle avoidance (for static obstacles). In the present section, the one-dimensional developments will be taken further to introduce a means for implementing robust time-optimal control with obstacle avoidance.

Optimal control may be represented in terms of the potential functions illustrated in figure 1. Using the techniques presented in Chapter 3, a feedback controller can be implemented which forces a system to conform to the idealized model of equation V.1 and the ideal behavior expected from the influence to the potential functions in figure 1. The controller of Chapter 3, however, only follows the optimal control potential functions; it is unaware of the influence of obstacle fields. To combine robust time-optimal control with obstacle avoidance, the concepts of a "minimally influential obstacle field", an "arrest point", a "limited lookahead window", and a "reflex subgoal" will be introduced.

A potential function will be defined as "active" at a point in space if it defines the value of U_{net}, as per equation V.3. The "minimally influential" potential field of an obstacle is the field corresponding to the potential function which: 1) can guarantee collision prevention without exceeding feasible actuator efforts; and 2) is "active" over the smallest possible region in joint space. In figure 1, the potential function U_{brake} satisfies the properties of a minimally influential potential function which protects a point obstacle at the position θ_{goal}. Since the slope corresponds to maximum braking effort, no feasible potential function may have a steeper slope. Also, since imposition of this function would result in the system coming to rest precisely at θ_{goal}, the obstacle field is influential only at the last possible moment, i.e., in the smallest possible region.

A useful property of minimally influential potential functions is that they do not occlude feasible goal locations. If the default control policy for target acquisition would not result in collisions with obstacles, then a minimally influential obstacle protection field will have absolutely no effect on the goal interception policy. This is an automatic result of the logical potential function combinations defined by equation V.3.

An arrest point will be defined for a system in terms of its state (ω, θ), and actuator saturation limits. For state (ω, θ), the arrest point is the point closest to θ at which the system can be brought to rest within actuator limitations. The set of points lying between the current position, θ, and the arrest point, θ_{arrest}, will be referred to as the "arrest region". For the simple system of equation V.1, the arrest point corresponding to the state (ω, θ) is:

$$\theta_{arrest} = \theta + \frac{\omega|\omega|}{2u_{max}/I} \tag{V.5}$$

The arrest point of a system has a simple relationship with respect to minimally influential obstacle fields. Under potential function control, when the minimally-influential potential field of an obstacle is active, the system arrest point coincides with the physical boundary of the active obstacle. That is, coming to rest as soon as possible coincides with coming to rest at the boundary of the obstacle.

An alternative interpretation of this observation is the following: if there are no obstacle boundaries present between a system's current position and the system arrest point, i.e., if no obstacles intersect the arrest region, then no obstacle potential field is active on the system. This fact may be employed to make potential function computations more efficient. For the computation in equation V.3, every potential function in the space of the manipulator must be evaluated at θ to determine which one is active. If all obstacle potential functions are minimally influential, then the arrest point may be used to reduce the computation. An inspection of the arrest region in configuration space may be performed to determine if a boundary of any obstacle lies within this region. If no boundary is found within this range, then no obstacle field is active. The nearest boundary encountered in this range corresponds to the boundary of a dominant active obstacle. Under conservative potential function control, no obstacle boundary should approach closer than the arrest point; obstacle repulsion corresponds to maintaining the arrest point precisely at the obstacle's boundary.

A search for obstacle boundaries in the range $\theta \rightarrow \theta_{arrest}$ is more effective than execution of equation V.3 when many obstacles are present. As more obstacles are added, more comparisons are required to evaluate equation V.3. If an obstacle map (e.g., as described in Chapter IV) is available, however, a forward search for the first obstacle boundary encountered is as fast (indeed, faster) in a crowded environment as in a sparse environment.

In effect, the computation of equation V.3 may be restricted to those potentials which correspond to obstacles within view of a "limited lookahead window." At most, it is only necessary to examine the arrest region to determine if any obstacle potentials act on the system. In fact, it is not necessary to examine the entire arrest region. Since (static) obstacles can only enter the arrest region by passing through the arrest point, it is only necessary to continuously examine a small neighborhood about the arrest point to sense the presence of obstacle avoidance fields. Choice of the inspection neighborhood defines the limited lookahead window. For a perfect system, the lookahead window shrinks to a single point: the arrest point. Under perfect potential function control with minimally influential obstacle fields, any obstacle potential which is active corresponds to an obstacle with a boundary precisely at the arrest point. In real systems, it is necessary to examine a finite neighborhood about the arrest point in order to accommodate model and control imperfections.

A critical nonideality of potential function control implementation is the computation time required to perform the evaluation of equation V.3, or equivalently, the time required to perform the neighborhood scan about the arrest point. During evaluation of the obstacle fields, the arrest point may be moving. The neighborhood which is examined about a sample arrest point should be large enough to contain the instantaneous arrest point, as well as all possible trajectories of the arrest point which can occur during the examination cycle. The rate at which the arrest point moves is:

$$\begin{aligned}\frac{d}{dt}\theta_{arrest} &= \frac{d}{dt}\left(\theta + \frac{\omega|\omega|}{2u_{max}/I}\right) \\ &= \omega + \frac{\dot{\omega}|\omega|}{u_{max}/I} \\ &= \omega + |\omega|u/u_{max}\end{aligned} \qquad (V.6)$$

Under maximum deceleration, $u = -\text{sgn}(\omega)u_{max}$, and thus $\frac{d}{dt}\theta_{arrest} = 0$. Therefore, under maximum braking, the arrest point is stationary. This is logically consistent, since the closest point at which a system can be brought to rest should remain the same as the system is brought to rest as quickly as possible. On the other hand, if the system is accelerating at the maximum feasible rate, $u = +\text{sgn}(\omega)u_{max}$, then the arrest point advances at the rate $\frac{d}{dt}\theta_{arrest} = 2\omega$. Therefore, if a map scan for obstacle boundaries requires a computation time of dt, the neighborhood to be scanned should include the anticipated arrest point at time $t + dt$. Thus, the width of the neighborhood to be inspected should be at least $2\omega dt$.

The neighborhood to be tested for the presence of obstacles corresponds to a lookahead window which extends from θ_{arrest} to $\theta_{arrest} + 2\omega dt$. In addition to scanning forward from θ_{arrest}, it is pragmatic to scan a limited distance backwards as well. It was shown in Chapter 4 that obstacles may be computed and stored in configuration space more efficiently in terms of their edges (or shells) than their complete ranges (volumes). Ideally, the edge of an obstacle remains coincident with the system arrest point whenever the obstacle's braking field acts on the system. Imperfect execution of potential function control, however, can result in the edge of an obstacle moving inside the arrest range, $\theta \rightarrow \theta_{arrest}$. If an obstacle's interior is not represented in configuration space, and if the interior of the arrest range is not inspected, then if the boundary of the obstacle enters the arrest range, the obstacle will not be discovered by inspection of the proposed limited lookahead window. Thus, the obstacle's repulsive fields would be incorrectly ignored. Even if the control implementation were perfect, an obstacle boundary could still enter the arrest range if the obstacle moves unexpectedly. In this event, the robot must detect the encroaching edge, and move away from the approaching obstacle. Since it is possible, in practice, for an obstacle boundary to enter the arrest range, the limited lookahead window should include a neighborhood about the arrest point which extends inside the arrest range.

A search for obstacle boundaries within a neighborhood of configuration space about a system's arrest point is an efficient means of determining whether an obstacle's potential function is active on the system. If an obstacle potential function is active, then the nominal control policy for time-optimal motion to the goal location must be aborted, and obstacle avoidance initiated. When an obstacle boundary is detected at the system arrest point, immediate braking with maximum effort will bring the system to rest before colliding with the detected obstacle. In practice, the arrest point may be computed with respect to a conservative estimate of the feasible braking effort, $\hat{u} < u_{max}$. Braking may then be implemented robustly using the controller described in Chapter 3. Using the conservative computation of the arrest point, some penetration of an obstacle edge into the arrest region can be tolerated and corrected before an actual collision occurs. The percentage of permissible edge invasion is equal to the percentage of effort overhead reserved. That is, collisions can still be prevented if: $|\theta_{obs} - \theta|/|\theta_{arrest} - \theta| > \hat{u}/u_{max}$.

If an obstacle edge is detected within the limited lookahead window, then immediate braking must be initiated. Immediate maximum braking is equivalent to suddenly assigning the obstacle edge location as the goal point for the controller of Chapter 3. Since this controller enforces (nominal) maximum braking as it approaches a goal, it effectively enforces the control prescribed by an obstacle's braking field.

It has been shown that enforcing optimal control potential functions as per figure 1 is equivalent to invoking the controller of Chapter 3 with a setpoint of θ_{goal}. Further, enforcing obstacle avoidance with minimally influential potential functions is equivalent to suddenly reassigning the setpoint to the edge of any obstacle which is observed within a limited lookahead window. The sudden reaction to an impending collision is heuristically similar to reflexes. The normal behavior (in this case, time-optimal motion to a goal), is unaffected by the presence of obstacles, unless obstacle avoidance measures are absolutely necessary. When danger of a collision is imminent, the "reflexes" override any commands to the motors and invoke their own evasion commands by substituting a new, safe setpoint. The reflexes do not attempt to optimize any criteria; they only act to assure safety. In this sense, the reflexes are not intelligent, yet they may assume a priority which overrides higher-level commands.

The sudden change in setpoints which is invoked by the reflexes upon encountering an active obstacle may be executed more smoothly and robustly without reducing the performance of the time-optimal controller. This may be accomplished through the use of reflex subgoals. A reflex subgoal is closely related to the arrest point. Under time-optimal control, whenever a goal setpoint is located outside the arrest region, time-optimal control will prescribe maximum acceleration towards the goal. Braking only occurs when the arrest point approaches the goal setpoint. If a time-optimal controller, e.g. that of Chapter 3, is given a moving setpoint, then the controller will continue to exert maximum acceleration in the direction of the goal as long as the goal lies outside the arrest region. This observation leads to the definition of a reflex subgoal.

Examination of the limited lookahead window includes an inspection of the region from $\theta_{arrest}(t)$ to $\theta_{arrest}(t + dt)$. If an obstacle is found within this region, then the control setpoint is set to the edge of the obstacle. If no obstacle lies within this range, then the goal setpoint may be set to a subgoal at $\theta_{arrest}(t + dt)$. Since this location is included in the inspection region, and since, by assumption, no obstacles lie within the inspection region, this location is known to be safe. Further, since this setpoint lies beyond the arrest point from time t until time $t + dt$, the system will continue to accelerate toward the subgoal at maximum control effort until the next inspection cycle. Maximal acceleration towards the subgoal, however, has the same effect as maximal acceleration towards the ultimate goal.

Thus, the proposed selection of reflex subgoals results in a sequence of setpoints which either achieves time-optimal control to the ultimate goal, or brings the robot to a safe halt at the edge of a blocking obstacle.

The use of reflex subgoals is ideally identical to suddenly switching setpoints from an ultimate goal to an obstacle edge when an obstacle becomes active. The use of reflex subgoals is more robust, though, since the robot is never given a command which is unsafe. Thus, if the reflex controller were to suddenly stop functioning, the last subgoal computed would remain as the robot's setpoint, and this setpoint is guaranteed safe and feasible (at least when the obstacles are not moving). The robot would come to a halt at the subgoal, and fail to progress further toward the ultimate goal. This type of failure is more desirable than the alternative. If incremental safe subgoals are not used, then a failure of the reflex controller will result in the robot proceeding at maximum speed towards the goal, regardless of obstacles in its path.

Reflex subgoals are also valuable for automatically restricting the speed of the robot to a bound within which the reflex controller is competent. As the speed of the robot increases, the reflex controller must examine a correspondingly larger neighborhood about the arrest point. If incremental subgoals are not used, then the robot may continue to accelerate until its velocity becomes so large that the reflex controller is incapable of keeping up with the moving arrest point. Consequently, the robot may penetrate active repulsive fields before the reflex controller has time to exert their influence on the robot, resulting in a collision. Collision avoidance is thus only guaranteed if the reflex controller is known to be capable of examining an entire critical lookahead window before the system arrest point exits the window region. With the use of reflex subgoals, the arrest point can never escape the most recent neighborhood which has been fully inspected. As the robot's velocity grows, if the reflex controller is incapable of scanning ahead faster than the motion of the arrest point, then the system will begin to decelerated towards the last subgoal computed by the reflex controller. Deceleration will reduce the robot's velocity until the velocity of computed subgoals equals the velocity of the robot. Thus, the velocity of the robot is automatically constrained by reflex setpoints to keep the dynamics of the robot within the competence of the reflex controller.

To this point, the analysis of potential functions and their invocation in terms of sliding-mode optimal control and reflex control has been treated only in one dimension. For a dynamically decoupled system, sliding-mode optimal control may be implemented separately on each subsystem. The concepts of minimally influential potential fields, the arrest point, the limited lookahead window, and reflex subgoals generalize to higher dimensions, with some additional considerations. Potential functions will first be reconsidered in higher dimensions, which will lead to a generalization of reflex control.

§5. Potential Function Control in Higher Dimensions

In the preceding one-dimensional analysis, several important observations and definitions were made. For dynamically decoupled systems, e.g. the robot described in Chapter 2, it is tempting to try to extend the one-dimensional results to higher dimensions by applying the one-dimensional definitions and procedures to each degree of freedom independently. At the level of motor control, such an approach is feasible, since, in a dynamically decoupled system, the position, velocity and acceleration of any joint variable do not affect the dynamics of any other joint variable. For obstacle avoidance, however, coordination of the individual control systems is required.

The dynamics of an arbitrary dynamically-decoupled, frictionless system can be expressed as a generalization of equation V.1:

$$\begin{aligned} \mathbf{I}\ddot{\vec{\theta}} &= \vec{u} \\ \text{where} \quad |u_i| &\leq u_{max,i} \end{aligned} \tag{V.7}$$

For a dynamically decoupled system, the inertia matrix, **I**, consists of constants along the diagonal. Thus, each link satisfies equation V.1. The total system kinetic energy may be defined as a sum of energy components associated with individual links. The kinetic energy associated with link i, T_i, is defined as:

$$T_i = \mathbf{I}_{i,i}(\omega_i)^2/2 \tag{V.8}$$

Obstacle attraction fields which perform bang-bang time-optimal control may be constructed independently along each axis, following the same procedures as the 1-dimensional case described earlier. For each axis, the potential function slopes are set by the respective link actuator limits. The peak energy of each braking potential along each axis is set to zero. The energy of an attractive function along any axis is set to the negative of the kinetic energy associated with the respective link. The fields assembled in this manner will, in the absence of obstacle fields, drive the system to a goal position in minimum time. In the absence of obstacles, potential functions constructed in this manner and assembled for each axis according to equation V.3 will result in fields which drive the system to a goal location in minimum time. Thus, construction of potential functions for the realization of time-optimal control in higher dimensions is no more difficult than in the one-dimensional case.

Construction of obstacle fields in higher dimensions is not as simple as the construction of target fields. The objective of a target field is to drive the manipulator to a specified point on each axis and hold it there until all orthogonal goal coordinates are achieved simultaneously. For obstacles, however, the manipulator should avoid a specified region in space, which is equivalent to avoiding simultaneity of link positions which lie within configuration space obstacles. That is, target interception requires interception along all axes, whereas obstacle avoidance only requires avoidance along at most one axis.

An illustration of the obstacle potential field approach in two dimensions is given in figure 3. A rectangular obstacle (in configuration space) and a goal position are plotted in θ_1, θ_2. The manipulator is defined to start at the origin with zero initial kinetic energy. Attractive, braking and obstacle fields for both directions are constructed independently according to the rules for the one-dimensional

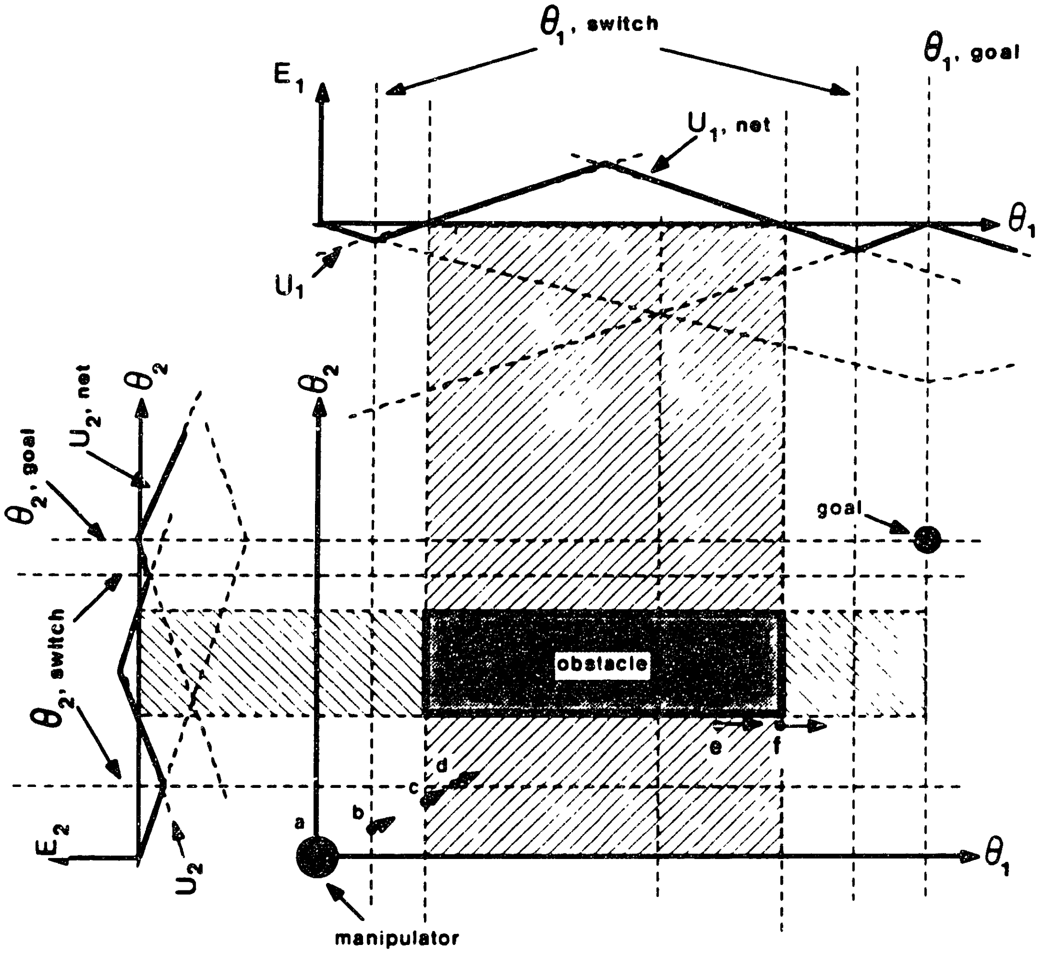

Figure 3: Potential Function Obstacle Avoidance in 2-D

case. The corresponding energy vs θ_1 and energy vs θ_2 plots are shown adjacent to the θ_1 vs θ_2 plot. The obstacle boundaries define strips in the θ_1 and θ_2 directions, the intersection of which comprises the obstacle area. In addition, dashed lines parallel to the θ_1 and to the θ_2 axes are extended from the switching points; these lines correspond to potential valleys of the energy plots.

If all potentials were constructed as shown, then obstacle safety would be guaranteed. However, the manipulator would be forever confined to the lower left region bounded by the obstacle boundary extension strips. Such an approach is grossly conservative, and prevents any solution to the simple target acquisition problem. Therefore, instead of enforcing both θ_1 and θ_2 avoidance simultaneously, avoidance in either θ_1 or θ_2 alone should be performed. This may be accomplished by selectively suppressing the influence of obstacle potential fields.

For purposes of selective field suppression, it will be useful to define three categories of potential functions. The three categories are defined with respect to a "surplus energy", $S_{i,j}$, for each axis i of each obstacle j. The surplus energy of the potential function along axis i associated with obstacle j is defined with respect to the state of the manipulator as:

$$S_{i,j} = U_{i,j}(\theta_i) + T_i(\omega_i) \tag{V.9}$$

where T_i is the kinetic energy of the ith axis of the manipulator. Equation V.9 assumes that the zero-energy level of U has been defined such that the rest energy of each link is 0. With respect to zero rest energy, conservative potential function control will maintain the relation: $U_{i,net} + T_i = 0$. Thus, if $S_{i,j} = 0$, then $U_{i,j} = U_{i,net}$. A potential function which satisfies $S_{i,j} = 0$ at a point defines the value of $U_{i,net}$ at that point. A potential function which defines a component of U_{net} will be called a "binding" potential function.

According to equation V.3, the binding potential function at any point along an axis is the function with the greatest energy at that point. When a potential field is selectively suppressed, its potential function may be excluded from the set of candidate functions which determine U_{net} in equation V.3. A function which is excluded from consideration in equation V.3 will be called "suppressed". A function which is not suppressed will be called "active. "

Normally, the surplus energy of a potential field is negative, meaning some other potential function, (the binding function), is controlling the motion of the robot. The surplus energy of a component potential function is zero when it is binding. If the surplus energy associated with some component potential function is positive, it is an indication that the function has been suppressed. In suppressing a potential function, the manipulator will not respond to the respective field. The manipulator may then burrow into a potential hill rather than climb its slope. The result is an apparent surplus energy associated with the suppressed potential.

Any potential function with nonnegative surplus energy at some point will be called "exposed" at that point, since its energy equals or exceeds that of the binding potential. At points where the surplus energy of a potential function is negative, the function will be called "submerged", since its energy lies below that of the binding potential.

With respect to the definitions above, rules can now be stated for selective suppression of obstacle potentials. Below, a very restricted class of obstacles will be considered: obstacles in configuration space which are rectangular (or rectangular prisms, or hyper-prisms), and which have edge (surface, hypersurface) normals aligned with the axes in joint space. The example obstacle in figure 3 falls within this class. Such obstacles have particularly simple, nearly-decoupled potential function construction and suppression rules. In reality, it would be highly unusual for a real obstacle to satisfy these properties. As shown in Chapter 4, though, an arbitrary real obstacle may be expressed in discretized configuration space as an assembly of square pixel contributions. Thus, the aligned, rectangular obstacle in configuration space is a fundamental building block for assembling arbitrarily complex configuration space obstacles.

The simplest case of potential function suppression occurs when the manipulator is receding from a rectangular obstacle, j, along some axis i. Then, the potential function contribution $U_{i,j}$ should be suppressed, since collision avoidance for obstacle j does not require repulsion along axis i.

The preceding observation may be expanded. If a manipulator is receding from the ith face of a rectangular obstacle, then it is not necessary to exert any of the repulsive fields due to that obstacle. All potential functions of the obstacle may be suppressed. For a manipulator at position $\vec{\theta}$ with velocity $\vec{\omega}$, potentials for an obstacle j may be suppressed along all axes $i \in \mathbf{N}$, according to the rule:

$$\begin{array}{ll} \text{if} & \exists i \in \mathbf{N} \text{ such that } \operatorname{sgn}[(\theta_i - \theta_i^{obs})\omega_i] = 1 \\ \text{then} & \text{suppress } U_{i,j} \forall i \in \mathbf{N} \end{array} \tag{V.10}$$

In the above, θ_i^{obs} is the i coordinate of the obstacle face which has a surface normal parallel to axis i pointing towards the manipulator. Since the present discussion is restricted to rectangular obstacles aligned with joint-space axes, the specified face and its ith coordinate are unambiguous. Equation V.10 states that all potential functions of an obstacle may be suppressed if the distance between the manipulator and any face of the obstacle is increasing.

An even stronger observation may be made by generalizing the notion of a minimally influential obstacle potential. If the potential function of obstacle j along axis i, $U_{i,j}$, is not binding, then all potential function contributions from obstacle j may be suppressed. That is, if $S_{i,j} < 0$, then it is still possible to prevent a collision between the manipulator and obstacle j by exerting the field of $U_{i,j}$ along the i axis. Obstacle avoidance fields for obstacle j may be suppressed until action is absolutely required.

Obstacle avoidance fields are not absolutely necessary as long as there is at least one axis for which the obstacle's potential function is submerged. When an obstacle's potential functions are all exposed, then it is no longer safe to suppress the repulsive fields. It is not necessary, though, to activate all of the fields; one is sufficient. If a single potential field is activated to prevent collision, that field must not have a positive surplus energy, or it will be incapable of halting the manipulator before collision. At the moment when repulsive fields must first be imposed, one (or more) of the obstacle's potential functions will have zero surplus energy. This (or one of these) function(s) should be chosen as the binding function. All other repulsive fields of the obstacle may remain suppressed.

The above rule for field suppression of minimally influential obstacle potentials may be summarized in terms of the surplus energy of each axis $i \in \mathbf{N}$. For any obstacle j:

$$\begin{array}{lll} \forall l \in \mathrm{N} & \text{if} & S_{l,j}(\theta_l) < 0 \\ & \text{then} & \text{suppress } U_{i,j}\ \forall i \in \mathrm{N} \\ & \text{else} & \text{suppress } U_{i,j}\ \forall i \in \mathrm{N}, i \neq k \\ & \text{where} & k : \forall i \in \mathrm{N},\ S_{k,j}(\theta_k) < S_{i,j}(\theta_i) \end{array} \qquad \text{(V.11)}$$

Normally, all potential functions of an obstacle are suppressed. At most, a repulsive field along one axis, k, will be required, where the function chosen to be activated has the minimum surplus energy of all of the obstacle's functions.

Use of equation V.11 is illustrated in two dimensions in figure 3. The manipulator is represented as a point in configuration space starting from rest at position "a". Attractive and braking potentials which describe the obstacle-free time-optimal goal acquisition solution act on the manipulator. In addition, potentials U_1 and U_2 protect a rectangular obstacle. The obstacle functions are binding only within the regions bounded by the dashed lines which extend from the respective switch points on the θ_1 and θ_2 energy plots. At point "a", neither U_1 nor U_2 are exposed, so their influence is suppressed. The goal attraction field is binding at this point, which causes the manipulator to accelerate toward the goal. At point "b", the manipulator enters the region where U_1 becomes exposed. However, since U_2 is still submerged at this point, equation V.11 suppresses any influence from U_1; the goal attractor continues to accelerate the manipulator. A consequence of suppressing U_1 at this point is that the surplus energy S_1 will become positive, and it will no longer be possible to prevent the manipulator from entering the strip which outlines the borders of the obstacle along the θ_1 axis.

At point "c", the manipulator is entering the region corresponding to the projection of a θ_2 face on the θ_1 axis. While in this region, the manipulator must be prevented from entering the corresponding region along the θ_2 axis, since the intersection of the two projections comprises the interior of the obstacle. At point "c", though, U_2 is still not binding, so no evasive action is taken yet. At point "d", U_2 just becomes exposed. Since all other potential functions of the obstacle are also exposed (i.e. U_1 in this case), the function with lowest surplus energy, U_2, becomes binding. All other obstacle fields (U_1) remain suppressed. Braking in the θ_2 direction continues on to point "e", where all of the θ_2-axis kinetic energy has been absorbed just as the manipulator reaches the face of the obstacle. Motion along the θ_1 axis continues under the influence of the goal potential fields alone. Finally, at point "f", the manipulator clears the obstacle in the θ_1 direction. At this point, U_1 is suppressed in accordance with equation V.10, and U_2 is consequently suppressed by application of equation V.11. The robot is then free to head toward the goal unconstrained. Since the manipulator is headed away from the obstacle boundary from this point, all obstacle fields are suppressed for the remainder of the move.

The present analysis extends potential function control to higher dimensions while preserving the concept of a minimally influential obstacle field. To do so, obstacles considered were restricted to rectangular regions in configuration space with edges normals (face normals) parallel to the axes. If an arbitrarily shaped configuration space obstacle is represented as a collection of small rectangular obstacles, then the arguments presented here apply to each of the component rectangles. The creation of many individual obstacles, however, would make the evaluation of equations V.10 and V.11 inefficient. To make the approach practical, the concepts of an arrest point, a limited lookahead window and reflex subgoals will be extended to higher dimensions.

§6. Reflex Control in Higher Dimensions

In section 4, the evaluation of multiple obstacle fields in 1-D was made more efficient through the use of a limited lookahead window. This technique is all the more important in higher dimensions, where the number of fields to be considered quickly becomes unmanageable.

An arrest point in higher dimensions is a natural extension of the definition in 1-D. For a dynamically decoupled system, there is a unique point in joint space which corresponds to the minimum distance at which the system can be brought to a halt. This point, the arrest point, is a single-valued vector function of the system state. The vector from a robot's current position to the arrest point is comprised of components equal to the braking distance along each of the respective axes.

The arrest point and the current manipulator position define opposite vertices of a rectangle (rectangular prism) in joint space. The area (volume, hypervolume) within this rectangle (prism) is the generalized arrest region. As long as configuration space obstacles do not intersect the arrest region, collision avoidance can be guaranteed. The objective of reflex control in higher dimensions, then, is to prevent such intersections.

Under perfect potential function control in one dimension, only the arrest point needs to be examined to determine the presence of obstacle braking fields. In higher dimensions, only the borders (surfaces) of the arrest region would need to be examined. Obstacles which do not intersect the arrest region (including the border) and are not contained within the arrest region do not exert active potential fields on the system. This statement can be verified by considering the following: 1) the system can always be brought to rest within the arrest region (including the border); 2) a minimally influential obstacle potential function exerts its field only when braking is essential to prevent collision. Since the system can be contained within the arrest region, it is always possible to avoid collisions with obstacles which lie outside the arrest region. Thus, all obstacles outside the arrest region (and not on the border) do not require immediate evasive action. By (2), any such obstacle may have its repulsive fields suppressed until one of its boundaries approaches the arrest region. Thus, it is only necessary to continuously inspect a neighborhood about the boundaries of the arrest region to detect if obstacle evasion fields should be activated.

As in the one-dimensional analysis, the critical neighborhood to be examined depends on the rate at which the arrest point can move. During the cycle time

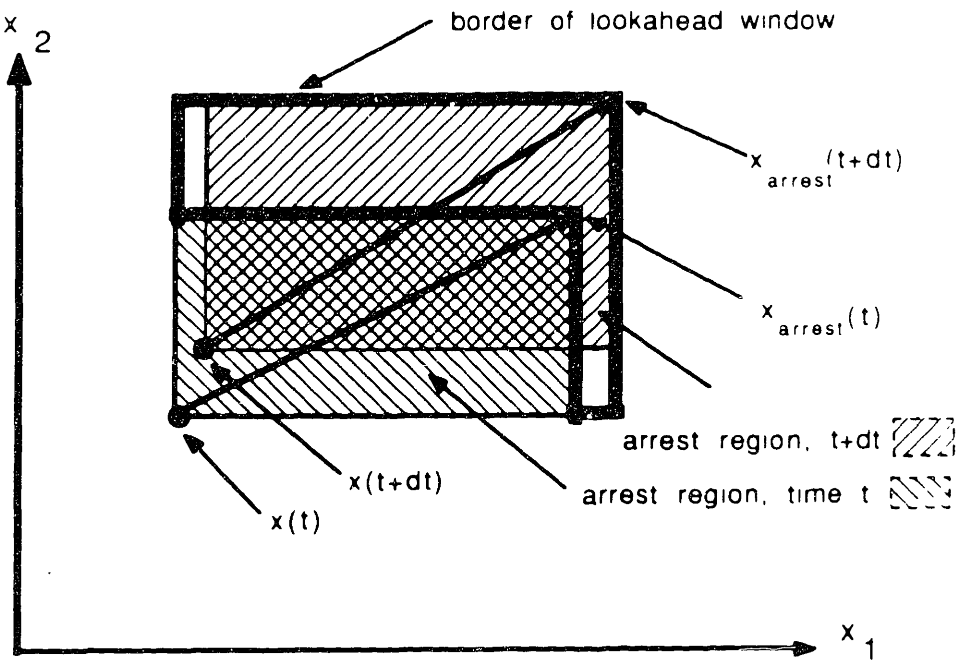

Figure 4: Limited Lookahead Window in 2-D

required to examine a neighborhood about the boundaries of the arrest region, the arrest point may move. To keep up with the motion of the robot, the neighborhood to be examined should contain all possible trajectories of the arrest point during the examination cycle time. The velocity of the arrest point is a vector in joint space with components described individually by equation V.6. To consider all possible trajectories of the arrest point from time t to time $t + dt$ it is sufficient to examine the area (volume,hypervolume) between the surfaces of the arrest regions at times t and $t + dt$.

An illustration of the critical neighborhood for examination in two dimensions is shown in figure 4. At the start of an examination cycle, the robot is at position $x(t)$. The examination cycle is assumed to take time dt, during which the robot may move as far as position $x(t + dt)$. The robot position at time t and the corresponding arrest point, $x_{arrest}(t)$, define opposite vertices of a rectangular arrest region. A worst-case rectangular arrest region for time $t+dt$ is also defined, based on the $x(t + dt)$ and the worst-case (most distant possible) $x_{arrest}(t + dt)$.

The area in the arrest region at $t + dt$ which is not within the arrest region at t corresponds to the maximum area which can be swept out by the edges of the arrest region as the arrest region evolves from time t to time $t+dt$. This swept area is the critical neighborhood, or limited lookahead window, to be examined by the reflex controller. During the examination cycle, the arrest point is guaranteed to be constrained to lie within the limited lookahead window. Any (static) obstacles which lie outside this window at time t will not have binding potential functions on the robot between time t and time $t + dt$.

Application of equation V.10 is implicit in the definition of the limited lookahead window. Points which lie within the lookahead window are contained within the regions swept by those edges (faces) of the arrest region which have normals in the direction of motion. Thus, the limited lookahead window does not consider faces which point away from the direction of motion. Obstacles near these faces will not be recognized, and thus their potentials will not be activated. Ignoring such obstacles has the desired effect of suppressing an obstacle's fields when the robot is receding from the obstacle.

If an obstacle is detected within the lookahead window, all of that obstacle's fields must be exposed. Immediate braking will bring the robot to a halt before the robot contacts the obstacle. It is not necessary, however, for the robot to brake simultaneously along all axes to avoid collision with any one rectangular (prismatic) obstacle; braking along at most one axis is required. The axis along which to brake is specified by equation V.11. In terms of the arrest region, the face along which the obstacle is first observed determines the braking axis. As braking is applied, the arrest region will shrink along the braking axis, though it may continue to grow along the remaining axes. Ideally, the face of the arrest region which first encountered the sensed obstacle will remain adjacent to the obstacle as the braking distance along this axis decreases. If the robot does not clear the obstacle in one of the other dimensions, its velocity along the braking axis will fall to zero as the robot approaches the boundary of the obstacle.

For a single obstacle, at most one axis of the robot will require braking to prevent a collision. When more than one obstacle is present, however, multiple obstacles can approach the arrest region simultaneously from different directions, in which case braking along multiple axes would be required.

As in the 1-D case, braking may be accomplished by suddenly assigning a new setpoint to the robot which is adjacent to the face of the sensed obstacle. For braking along the i axis only, only the i component of the goal setpoint must be affected; motion along the remaining axes may continue unaffected. In section 4, it was shown that incremental subgoals can achieve the same effect more robustly. Reflex subgoals also apply to higher dimensions. Any subgoal on or beyond a face of the worst-case arrest region predicted for time $t + dt$ will result in maximum acceleration towards that face throughout the time interval t to $t + dt$. A subgoal on a face of the arrest region at time t will induce maximum braking along the axis normal to that face. Braking only occurs as a subgoal enters the lookahead window. Subgoals are normally selected such that braking does not occur, i.e., by placing the subgoal at or beyond $\theta_{arrest}(t + dt)$. Braking along any one axis occurs only when an obstacle has entered the corresponding face of the lookahead window, or when the respective link nears its goal position.

Each component of a subgoal is selected such that: 1) the subgoal does not penetrate any obstacle; 2) each subgoal lies inside or on the boundary of the lookahead window; and 3) each component is as close as possible to the respective goal component, subject to the restrictions of 1) and 2). As a link of the robot approaches its goal location, the corresponding component of the subgoal will approach its respective goal component. At this point, the subgoal will lie within the lookahead window, and maximum braking will be exerted along the respective axis to bring the respective link to rest at its goal position. The face of the arrest region normal to this axis will remain stationary as the robot advances, and the respective dimension of the arrest region will shrink. As all of the links approach their respective goals, the arrest region will shrink to a point coincident with the goal. By selecting subgoals in this manner, time-optimal motion to a goal position may be preserved, provided obstacle avoidance is not required. When it is absolutely necessary to commence braking to avoid a collision with some obstacle, then braking is enforced by neglecting to advance the subgoal setpoint in the direction of danger. Neglecting to advance a safe subgoal when permitted may occur if the reflex controller fails, or if it is incapable of keeping up with the robot dynamics. Failure in this mode is tolerable, since the worst that can happen is that the robot will come to a safe halt.

The alternative approach, suddenly establishing a new safe setpoint at the moment an active obstacle field is detected, is more risky. Failure to keep up with the robot, or complete failure of the reflex controller will leave the robot unprotected.

In the present discussion, the generation of safe reflex subgoals has been described with respect to stationary obstacles. For static obstacles, a safe reflex setpoint remains safe for all time, and a new obstacle field may present itself for consideration only when a moving arrest region sweeps over the respective obstacle border. When obstacles move, however, a reflex setpoint is not guaranteed to remain safe for all time. If the robot remains at rest at a position which is initially safe, an obstacle may approach the robot, in which case the setpoint becomes dangerous and the robot must move away from the obstacle. Alternatively, the robot may be decelerating towards a feasible arrest point near the edge of an obstacle. If the obstacle then moves toward the decelerating robot, the arrest point to which the robot is headed will become unsafe. In both of these instances, an obstacle potential enters consideration by advancing towards the robot rather than via the lookahead window frontier advancing toward the obstacle.

In this thesis, rigorous generalization to moving obstacles has not been attempted. Consideration of quasi-static obstacles (those for which the configuration space obstacle borders move at speeds small compared to the robot) has been incorporated by augmenting the lookahead window. A "myopic" search of the configuration space map in the vicinity of the robot is performed to detect if obstacles are encroaching on the robot. Obstacle potentials detected within this extra search region are included in the determination of a reflex setpoint. With this addition, the robot is repelled by approaching obstacles (at least for low-speed obstacles). Further, the limited lookahead window is broadened somewhat by displacing the border at time t backwards to include obstacle potentials which may have penetrated the arrest region. For static obstacles, no obstacle borders should be able to penetrate the arrest region, though penetration may occur for moving obstacles. As discussed in chapter III, near-optimal control is implemented using an underestimate of the true actuator saturation, leaving a reserve effort for corrections. For obstacles which marginally penetrate the arrest region, the actuator effort reserve permits halting the robot at a safe reflex setpoint inside the arrest region.

Limited lookahead windows and reflex subgoals as described here have been implemented in two dimension for the planar robot described in Chapter 2. Implementation results are presented next.

§7. Implementation of Reflex Control in Two Dimensions

Reflex control was implemented for the 2-link dynamically decoupled planar robot described in Chapter 2 using the controller of Chapter 3 and the discretized configuration space representation described in Chapter 4. A limited lookahead window was used to compute reflex subgoals such that each subgoal was guaranteed safe, yet (typically) did not inhibit time-optimal control in free environments.

In discretized configuration space, examining a lookahead window consists of checking the state (on or off) of each pixel within the window. Using reflex subgoals, the examination window may be arbitrarily small; the robot may be servoed to incremental guaranteed safe subgoals as they become available. In order for the reflex controller to permit time-optimal motion, though, reflex subgoals must advance faster than the arrest region boundaries.

The size of the appropriate examination window depends on the velocity of the robot, the acceleration limits of the robot, and the cycle time of the reflex controller. For the robot described in Chapter 2, the maximum accelerations of links 1 and 2 are 275 rad/sec^2 and 960 rad/sec^2, respectively. The angular velocities of the links are limited by the back EMF of the motors and the available voltage of the PWM amplifiers. The velocity limits of the motors are approximately 3000 rpm, which corresponds to 78 rad/sec for link1 and 314 rad/sec for link2. At maximum acceleration, these velocity limits could be reached after 1.8 revolutions of link 1 and 8.2 revolutions of link 2. A move requiring multiple revolutions of the robot would be highly unusual. More realistic upper bounds on the link velocities can be estimated by considering the peak velocity achieved under bang-bang control during a move of one-half revolution. For link 1, the maximum velocity reached for a move of one-half revolution is 29 rad/sec. The corresponding peak velocity of link 2 is 55 rad/sec.

At the maximum expected velocity of the robot, equation V.6 for the maximum velocity of the arrest point evaluates to 58 rad/sec along axis 1 and 110 rad/sec along axis 2. In the chosen discretization of configuration space (0.05 radians per pixel along each axis), the worst-case velocity of the arrest point is 1180 pixels/sec along axis 1 and 2240 pixels/sec along axis 2. At these velocities, maximum braking would bring the links to rest in a braking distance of $\pi/2$ radians. Thus, the arrest region has dimensions $\pi/2$ by $\pi/2$, or 32 pixels by 32 pixels. Only two edges of the the arrest region must be scanned: those edges with outward normals in the direction of motion. A border scan 1 pixel deep would require a test of 65 pixels. Link 2, the higher velocity link, advances its arrest point component by 1 pixel in 446 microseconds at maximum velocity. For the reflex controller to scan forward faster than the velocity of the arrest point, the test of all 65 pixels must be complete within 446 microseconds, which corresponds to a pixel test rate of nearly 146 kHz. The reflex controller as implemented on the MC68020-based single board computer tests pixels at 48 kHz. Thus, the reflex controller does not permit the robot to move as quickly as possible. For long moves, the reflex subgoals are updated more slowly than the velocity prescribed for optimal control.

In the reflex control implementation, the lookahead window has been defined to include a search of pixels along the two leading arrest region borders with a search depth of 6 pixels in the θ_1 direction and 8 pixels in the θ_2 direction. In addition to searching the borders which propagate outwards, an additional 24 pixels are searched about the arrest point in the backwards direction, to test for any obstacles which may have penetrated the arrest region. This additional search handles cases of (low speed) moving obstacles which approach the robot (as discussed in section 6), as well as nonidealities in the controller which can permit slight penetration of the nominal arrest region by static obstacles. Further, a search of all pixels surrounding the current robot location is performed to determine whether moving obstacles are approaching the robot.

At high velocities, the lookahead window grows to an area which can not be fully inspected in the time required for transparent reflex control. The robot is always constrained to move only within known safe regions, but the rate at which regions are known to be safe may be discovered too slowly to accommodate the full velocity capacity of the robot. An example is illustrated in figure 5. In this figure, the arrest region shown corresponds to that which would result from maximum acceleration of both links for 42 ms. The velocities after such an impulse would be 11.5 and 40.3 radians per second for links 1 and 2, respectively. The corresponding dimensions of the arrest region are 5 pixel widths and 17 pixel widths, respectively. A search along the borders 6 pixels deep in θ_1 and 8 pixels deep in θ_2 consists of 214 pixels (including the 24 tests for invading obstacles), which can be performed in 4.4ms with the current realization.

Figure 5 indicates the order for testing pixels. First, a vertical scan of all pixels adjacent to the arrest region border at θ_1^{arrest} is performed. This set of pixels is denoted by an arrow labeled "1" for the first row scan. Each pixel in this range is tested for the presence of an obstacle face with a surface normal

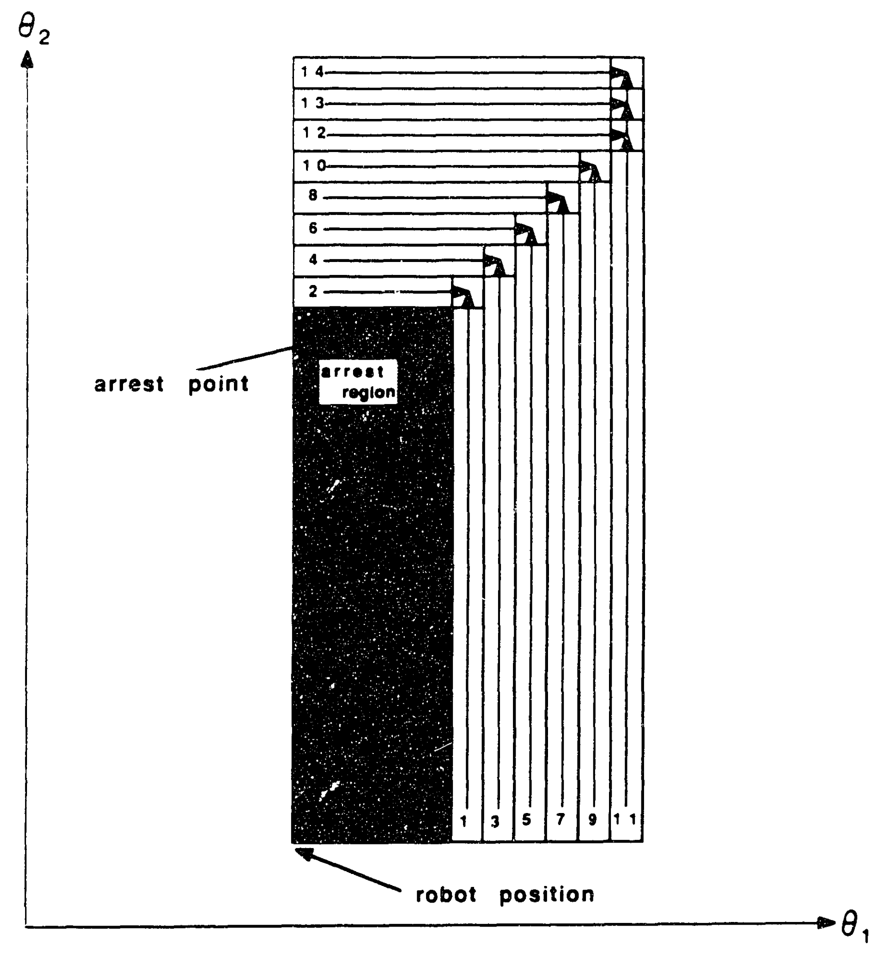

Figure 5: Window Inspection Implementation in 2-D

pointing along the negative θ_1 axis. If any such face is found, the θ_1 component of the next subgoal is not permitted to advance beyond the respective component of the current arrest point. If an obstacle were discovered during this scan, no further vertical pixel scans would be performed in this cycle. Further tests for permitted motion along the θ_2 axis may, however, continue. A horizontal scan is performed next for pixels along the horizontal border of the arrest region. This next row scan is labeled "2". Each pixel in the row is tested for the presence of an obstacle face with a surface normal pointing along the negative θ_2 axis. If any such obstacle is found, the θ_2 component of the next subgoal is restricted to the respective component of the arrest point.

Alternate vertical and horizontal row scans are performed until: 1) the entire window has been inspected (less regions blocked by known obstacles), or 2) obstacles are encountered along both axes, or 3) the goal position is reached within one of the inspected pixels. For each axis, the search is not continued beyond an obstacle wall or beyond the goal position of that axis. After a complete search, a subgoal is chosen which lies within the inspected region as close to the ultimate goal as possible. In figure 5, if no obstacles appear within the lookahead window, the subgoal will advance to 6 pixels beyond the θ_1 arrest point component and 8 pixels beyond the θ_2 arrest point component.

A complete search of the lookahead window of figure 5 would require inspection of 214 pixels (including 24 pixel tests for obstacles within the arrest region). In the current implementation, the region could be inspected in 4.4ms. During a cycle of 4.4ms, under continued maximum acceleration the robot would advance the arrest point 2 pixels along θ_1 and 7 pixels along θ_2. Since the subgoals are selected to lie beyond the maximum motion of the arrest point during the examination cycle, the robot will continue to accelerate towards the subgoals. Thus, in this case the use of a subgoal does not limit the time-optimality of a move to the ultimate goal. Beyond 42 ms of acceleration, though, the time required to examine the correspondingly larger lookahead window will result in subgoals being delivered less frequently, and the robot will automatically limit its velocity as it begins to converge on each subgoal. The velocity of link 2 will be limited to approximately 40 rad/sec. Subgoal setpoints for link 1 will continue to be placed in advance of the arrest point along this axis until link 1 reaches a velocity of 15 rad/sec.

The speed of the reflex controller limits the velocities of the links. If only one link is accelerated at a time, then the arrest region will be longer and narrower, resulting in a lookahead window which encompasses fewer pixels than the previous example. As a result, the reflex controller permits higher link velocities when the links move individually than when they move move simultaneously at high speed.

Moving obstacles have not been treated rigorously here. Quasi-static obstacles are accommodated through the extensions of the lookahead window discussed above. A quasi-static obstacle may be defined in terms of the control bandwidth of the arm and the width of a safe "buffer" region which expands the defined configuration space outline of a moving obstacle. In the present implementation, low-speed repulsion from obstacles falls within the near-linear region of the controller described in chapter III, which employed critical damping and a closed-loop bandwidth of 10Hz. In the discretization of configuration space, as described in chapter IV, the chosen resolution was 128 pixels per link revolution. Obstacle outlines were computed with a safety region one pixel deep. Thus, for the implemented controller and obstacle representation, a setpoint which advanced at 1.5 rad/sec (31 pixels per second) would result in a position error of one pixel. In the current implementation, obstacle outlines are recomputed in the worst case at 30 cycles per second, which is appropriate for obstacle advances of 30 pixels per second.

Obstacles for which the configuration space border moves at 30 pixels per second or less may be considered quasi-static in the current implementation. For obstacles near the elbow (joint 2) of the robot, 30 pixels per second corresponds to a tangential motion of about 18 cm/sec. However, for obstacles which approach the robot's elbow in the radial direction, discontinuities in the configuration space transformation result in the sudden appearance of new forbidden regions. Such obstacle motions can not be considered quasi-static. A more rigorous treatment of moving obstacles would include an anticipation of obstacle motions in task space, and corresponding swept representation of configuration space transforms, as mentioned in Chapter IV, section 7. This extension has not been investigated here.

Examples of the reflex control implementation are shown in figures 6 through 10. In each case, the robot is standing still, so the arrest point coincides with the robot's current position. A window size of 6 by 8 pixels is inspected in the direction of the goal, plus additional tests in the vicinity of the robot to check for approaching obstacles. A single obstacle (in this case, a 2" circle at $x = y$

= 22.3 cm) is present in the robot's workspace, the edges of which transform to two closed regions in configuration space. In discretized configuration space, the single physical obstacle is represented as a collection of many individual square obstacles.

In figure 6, the goal lies within the inspection window, so the subgoal is chosen coincident with the ultimate goal. In figure 7, the goal lies outside the window. No obstacles lie within the window, so a subgoal is chosen at the lookahead window boundary, the closest known safe approach toward the goal. In figure 8, the goal is blocked by the obstacle in the θ_1 direction, but not in the θ_2 direction. A safe subgoal is assigned which allows progress along the θ_2 axis, but limits motion in the θ_1 direction to the position of the closest obstacle wall encountered. In figure 9, both axes are blocked; motion is permitted only up to the border of a known safe rectangular region. In figure 10, the obstacle has moved into the robot (or control imperfections have permitted the robot to enter the obstacle area). A safe subgoal is assigned outside the obstacle region.

The implementation described here prevents the robot from entering obstacle regions, even when the robot is moving at very high speeds. Reflex control is nearly transparent, except at such high velocities that the reflex controller is not capable of inspecting the correspondingly large lookahead window within the time that the robot's arrest point is capable of escaping the window. In that event, the velocity of the robot is automatically limited by failing to update subgoals at a fast enough rate. Thus, the reflex controller limits the speed of the robot to levels for which the reflexes are fully competent.

Reflex control may be implemented on parallel processors, though that was not done in the present implementation. A single processor was adequate for impressively high speeds. The worst-case performance of the reflex controller (slowest inspection rate) occurs when no obstacles are present. In very cluttered environments, the reflex controller completes its window inspection in less time than in obstacle-free environments. Thus, reflex control is well-suited for complex

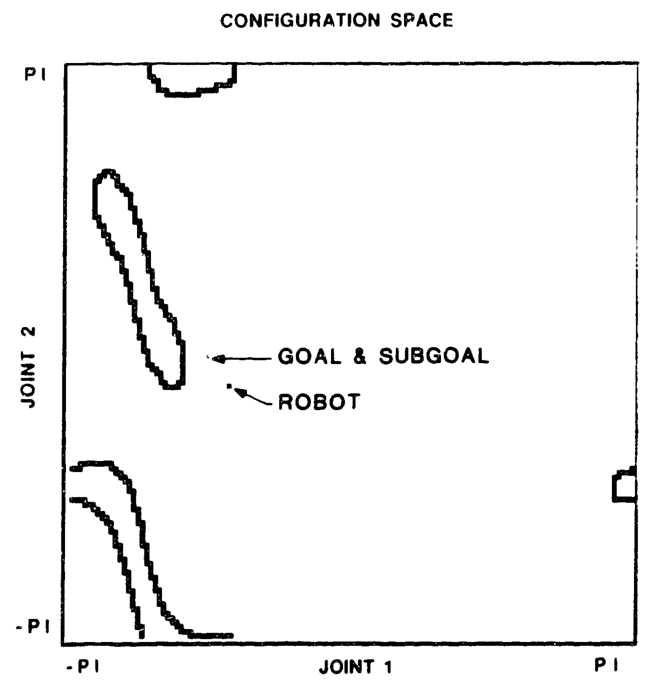

Figure 6: Safe Goal Within Lookahead Window

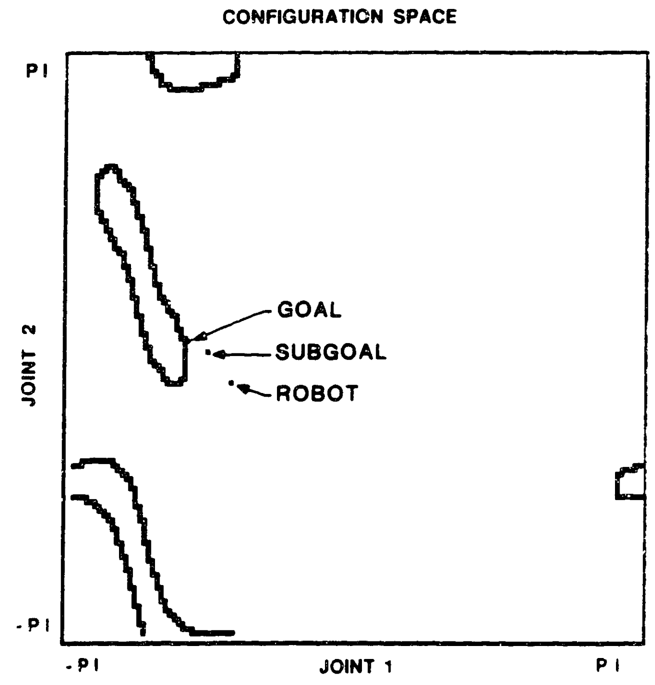

Figure 7: Safe Goal Outside Lookahead Window

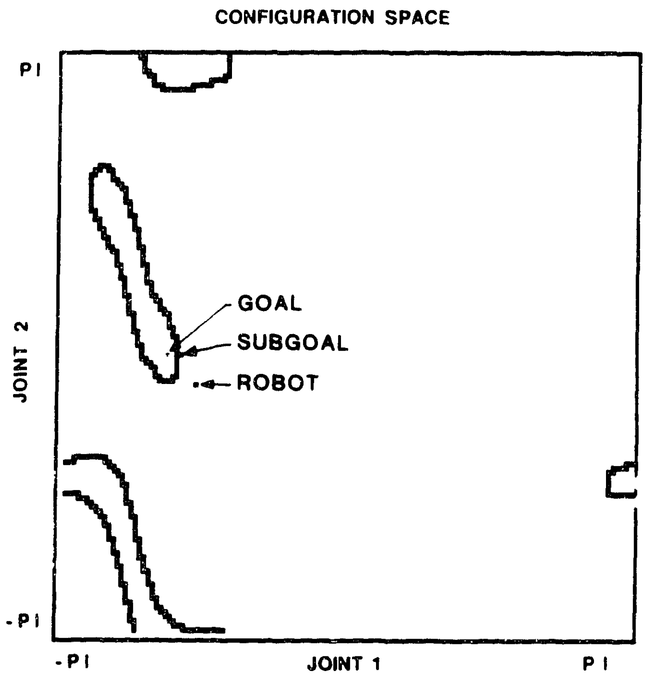

Figure 8: Goal Blocked Along One Axis

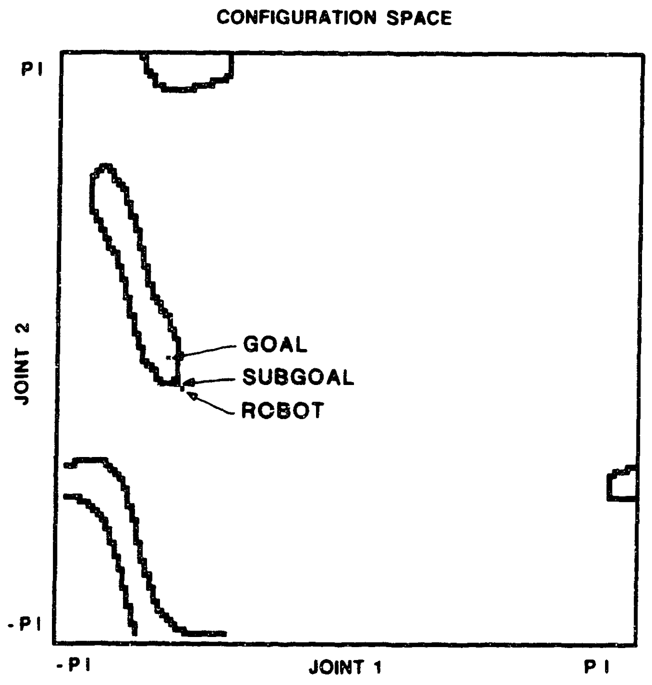

Figure 9: Goal Blocked Along Both Axes

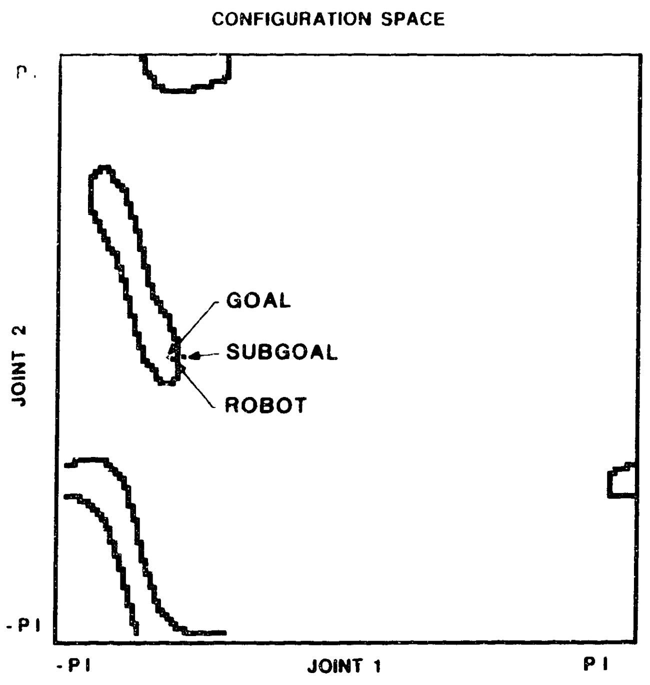

Figure 10: Escape from Obstacle Interior

systems. In higher dimensions, high speed competence would require faster computation or the use of parallel processing. In two dimensions, a single processor was deemed adequate.

§8. Summary and Conclusions

In this Chapter, reflex control was presented as a means for reconciling high-speed motion and guaranteed obstacle avoidance for robots in cluttered environments. Combining the techniques of sliding-mode time-optimal control, a discretized configuration space representation of obstacles, and variations on potential function control resulted in the derivation of reflex control. Reflex control utilizes the concepts of an arrest point and arrest region to define a limited lookahead window: a subset of configuration space which contains all information necessary to determine if continued high-speed motion is guaranteed safe. Obstacles are defined in terms of potential fields which are minimally influential; the potential fields have no effect on the robot dynamics unless absolutely necessary to prevent a collision. Actuator saturation is explicitly considered in the construction of the potential functions, so that collision avoidance is guaranteed theoretically, as well as guaranteed to be realizable, at least for static obstacles.

Reflex control is defined generally in arbitrary-dimensional spaces. Implementation was performed in 2-D. In the present implementation on a single processor, reflex control did limit the maximum speed capability of the robot, albeit only at very high velocities. In higher dimensions, parallel computation of reflex control may be necessary (depending on the speed capabilities of the robot).

In the present development, only static and quasi-static obstacles were treated. Slowly moving obstacles were considered by including a local search of configuration space about the position of the robot. As obstacle fields advance toward the robot (at low speeds) the robot was repelled away from the approaching obstacle. Rigorous generalization of moving obstacle avoidance for higher-speed obstacles remains a topic of future research.

Although reflex control permits high-speed motion and guarantees static obstacle avoidance, it is relatively unsophisticated in its analysis. No planning is performed, and only obstacles within the lookahead window are considered. Since reflex control is based on potential function control, it suffers from the same inherent limitations: paths are not globally optimized, and local energy minima can result in bringing the robot to a full stop, although a valid path to the goal may exist. In this sense, reflex control is analogous to reflexes. The reflexes are normally transparent (do not affect planned motions). They act on sensory data with only crude analysis, but do so very rapidly. Although the reflexes are less sophisticated in their analysis than higher levels of abstract reasoning, they have the capacity to override commands from higher levels and assume direct control over motor activities.

Reflex control is (intentionally) not competent to handle tasks requiring complex reasoning (e.g., maze solving). Thus, reflex control is not sufficient in itself to direct an effective autonomous machine. Integration with higher levels of control is necessary. In the next chapter, an example of integration with path planning is detailed.

CHAPTER VI

INTEGRATION OF REFLEXES WITH PLANNING

§1. Introduction

Developments to this point in the text have led up to a control structure which satisfies many of the features necessary for a competent autonomous system. The choice of joint space as a common reference frame for both control and representation of obstacles permitted the realization of both high-speed motion and guaranteed obstacle avoidance through the use of reflex control. Nearly time-optimal motion is achieved in unobstructed environments. In environments in which obstacles invalidate the default (obstacle free) control policy, the reflex level of control automatically prevents collisions, and, for simple obstructions, deflects the path of the robot into one which successfully reaches the goal. In complex environments, however, reflex control alone is not adequate to steer the robot to its goal. A more sophisticated strategy is required than simple repulsion from obstacles. The realization of strategies for effective motion in complex environments falls within the domain of path planning.

It should be realized that planning collision-free motions for an articulated manipulator is considerably more difficult than planning motions of an end-effector alone. For example, planning motions for a sphere among a field of obstacles seems simple and intuitive. However, planning motions for the same sphere attached to a kinematic chain is much more difficult when the links must avoid collisions as well. Path options for the kinematic chain are dramatically restricted with respect to the valid options of the sphere alone. The comparison is graphically apparent in configuration space, in which each point of a task space obstacle corresponds to a sprawling forbidden region (or regions) in configuration space. Having performed the transformation into configuration space, however, the complexity is incorporated in the shapes of the forbidden regions; the planning problem in configuration space may be treated identical to the problem of navigating a point automaton in Cartesian space.

In this chapter, a brief analysis of path-planning techniques is presented in which the advantages and limitations of local and global path planning are discussed, particularly with respect to sensory-interactive real-time execution. Generalizations from lower levels of control are drawn to deduce properties expected of additional levels of control. As a result, it is proposed that local planning and global planning be executed simultaneously, with local planning acting as an intermediate layer between reflexes and global planning. One particular local planning technique is selected and modified for use in dynamic environments. Implementation results and coordination details are presented for an integrated control system which was realized for the 2-D planar robot of Chapter II.

§2. Path Planning Techniques

In the area of path planning, a wide variety of assumptions and techniques have been explored. Techniques employed are generally algorithmic or heuristic. Algorithmic solutions involve a sequence of operations which can be formulated to guarantee the discovery of a path solution (guaranteed convergence) whenever a solution exists. Guaranteed convergence, however, translates into the equivalent of an exhaustive search in sufficiently complex cases. For an exhaustive search to be feasible, the set of candidate paths is restricted, e.g. by discretizing the search space [40], or by constraining the path to a sequence of line segments which join a finite selected set of points in space [25].

An alternative to algorithmic planning is a heuristic search, in which a general (though not infallible) rule is used for making decisions about which paths to explore based on some evaluation of the apparent merit of the various options [e.g., 15]. A heuristic technique may involve assumptions about the problem, and choose options which often result in faster discovery of a solution than an algorithmic approach. However, heuristic approaches do not generally guarantee optimality, or even convergence to a solution.

A second major categorization of path-planning literature involves the set of assumptions employed. Path-planning is most commonly approached with the assumption that all obstacle boundaries are known a priori, e.g. in the "piano mover's problem" [43,44]. This category of planning will be referred to as "global planning", since complete knowledge of the world is used. Another class of problems has been investigated in which only limited knowledge of the environment is assumed [e.g., 28,29,30], such as in the case of a rat running an unfamiliar maze. For this category of problems, obstacle information is available only within a limited range about the current position; planning in this category will be referred to as "local planning."

Local planners can not guarantee optimality in the sense of minimum path length. Paths resulting from planning algorithms using only limited knowledge of the environment can be extremely inefficient in comparison to paths optimized with respect to full knowledge of the environment. However, path planners which utilize full information are typically slow in obtaining solutions [14,15,40], requiring from minutes to hours of computation time, even for simple problems. In [15], a heuristic planner in seven dimensions computed non-optimized paths in 7 hours on a MicroVax.

In addition to being slow, global path planners are also sensitive to changes in the environment. If the goal location changes, or if the placement of any obstacle which affected the computed path changes, then any path computed prior to the displacement is invalidated. If the goal state is moving continuously and/or if obstacles are moving about, then virtually all current path planning approaches are not applicable. An exception to this generalization is [17], in which it is assumed that the motion of obstacles is known a priori; path planning is performed in configuration space augmented by the dimension of time. Assumptions about complete knowledge are even stronger in this approach, however, since complete knowledge of the future (of obstacle boundary locations) is required. Further, the extra dimension of time increases the computational burden of an already complex problem. Solutions obtained are only valid if initiated at a precise moment in time when the environment conforms to the assumed initial conditions. Such an approach is not consistent with sensory interactive control, since solutions obtained would not be valid at the moment they were needed.

A limitation of both local and global path planning is that they do not take robot dynamics into consideration. Optimization of a path is typically defined as minimization of the path length, even though a path of minimum length is not typically a path which can be followed quickly. For example, the minimum-length-path to a goal which is obstructed by polygonal obstacles consists of a sequence of line segments with endpoints at obstacle vertices and/or the start and goal points [25]. Although such a path truly minimizes distance, the requirement of turning a sharp corner at each line segment endpoint results in a severe time penalty, since the system must come to a complete halt at each corner.

Trajectory planning, a generalization of path planning, considers both the path of the robot and the time history of positions along the path. Trajectory optimization is considerably more difficult than path optimization, since the trajectory search space is at least twice the dimension of the path search space. Even in obstacle-free environments, trajectory optimization is formidable. Solution attempts generally assume simplified robot dynamics [34], simplified constraints [45], or a coarsely discretized search space [40]. A more restrictive assumption still is that the full path is given a priori, and minimum-time motion along the path is computed [8]. In [14], an iterative technique for finding approximate time-optimal trajectories is described in which maximum speed along specified paths is computed, and successive trial paths are varied parametrically. In general, the consideration of dynamics in the computation of optimal paths is so computationally intensive that its application in sensory interactive systems is far from practical.

A viable autonomous system must be capable of responding instantly to a changing environment, should exhibit effective use of its full dynamic capabilities, and should utilize global information in deducing efficient motion strategies. These requirements are not consistent with any of the methods described so far. Global path planners are capable of deducing intelligent paths, but are too slow and too sensitive to changes in the environment. A system which depended on a global path planner would be stalled until a path solution was available. In dynamic environments, any computed path solution would be invalidated before it could be executed, resulting in total paralysis of the system. Since a local planner uses only limited information, it has the potential to respond rapidly to local changes in the environment. Decisions using only local information, though, cannot exhibit much intelligence. Further, neither local planners nor global planners are competent in considering robot dynamics. However, a system which coordinates concurrent processes of global planning, local planning and reflex control has the potential to combine the desirable features of each technique.

In the following, it will be shown how local planning may be used as a bridge between reflex control and global planning in the construction of a control system which exploits advantages of each level. In this thesis, investigations in control integration stop at the local planning level. Ultimately, global planning as well as higher levels of control should be integrated with the present levels of high-speed servo control, reflexes and local planning, but no higher levels are considered here.

§3. Integration of Logical Processes in Real-Time Control

Any digital control system may be thought of as a decision-making process, the result of which influences the behavior of the controlled system. At the motor control level, the problem domain may be described formally (e.g., with Z-transforms) and control decisions may be expressed compactly in terms of a functional evaluation (the digital control law). Decisions must be made very rapidly at this level for the controller to be successful, where the decision rate (sampling rate) should be significantly faster than the characteristic time scale of the system to be controlled. These features of digital control system design are well formalized and well understood. The application of more general logical operations in real-time control, however, is not formalized. Current use of deductions, expectations, inferences, searches, heuristics and various A.I. techniques in control is ad hoc at best. Some observations from control theory will be made and inferences drawn to postulate properties which should be present in controllers which utilize non-traditional control laws.

The lowest level at which logic has been introduced in the experimental robot control system described here is quite mundane. The pulse-width-modulated amplifiers themselves utilize hardware logic to control the currents in the motors. The use of logic is especially primitive, but correspondingly fast: positive voltage duty cycle is increased if the current is too low, and decreased if the current is too high, subject to pre-defined current saturation limits. The objective of the PWM controller can be expressed simply: regulate the current. Control decisions are made at a rate of 5 kHz. Higher switching frequencies are possible, but would not result in noticeably better performance. Slower switching speeds, however, can be a problem. Lower switching speeds would result in poorer regulation of the motor currents and, at lower rates still, instability. Minimum decision rates are dictated by the time constants of the electrical dynamics. The PWM controller will fail only due to amplifier or motor component failures.

Somewhat more sophisticated logic is employed in the implementation of sliding-mode optimal control. At this level, anticipation is incorporated with a functional description, and amplifier commands are modulated based on an evaluation of that function. Further, a trivial form of drawing inferences by a combination of sensory input and deductions from a world model is present in the implementation of velocity observers. Still, the objective of the controller can be stated simply: regulate the function $\tilde{s} = 0$. Control decisions are made at over 750 Hz (roughly 1.5kHz/link, both links controlled by the same CPU). A more rapid response would not degrade performance but would not improve performance either, since the PWM amplifiers would not be able to respond to higher frequency commands. The decision rate may not be much slower, though, or regulation of $\tilde{s} = 0$ will be poor, and at low enough rates the controller will be unstable. The minimum decision rate is set by the time constants of the mechanical system, which are longer than those of the electrical system.

This servo-level of control would be invalidated by any failure of the lower level, PWM current regulation. Failure of the sliding-mode controller, however, would not affect the proper operation of the PWM controllers. Currents would continue to be regulated, and would be restricted to lie within the specified current saturation bounds, although the commands to the amplifiers may cease to make sense. The consequence of failure of the sliding-mode controller is possible mechanical damage to the robot.

At the reflex level, more formal logic is introduced in order to anticipate and respond to impending danger. The logic employed, though, is simple enough to be expressed in terms of primitive set operations on potential functions. The task description is more abstract than the previous two levels, but can still be described in simple logical terms: establish subgoals which are as close to the requested goal as possible within a restricted window and external to all obstacles within that window. In the current implementation, subgoals are generated at rates between 200 and 600 Hz. A faster rate of subgoal generation would not hurt system performance, but as long as the reflexes are capable of leading the arrest point of the mechanical system, faster subgoal generation rates would do nothing to improve system performance. A lower rate of reflex subgoal generation would result in degraded performance, though. If subgoals are not generated fast enough to lead the system arrest point, then the maximum velocity of the system will be limited. At still lower rates of subgoal generation, jerky motion would result, since the sliding-mode controller would bring the robot to a halt at each successive subgoal location. Further, if obstacles are moving, then the capacity for reflex subgoal propagation must be faster than the rate of advancing obstacle frontiers, or the reflex controller will not competently cope with the changing environment.

Of course, any failure of the lower levels (PWM amp control and sliding-mode control) will invalidate the reflex controller. Failure of the reflexes, though, is not physically disastrous (barring attacks by moving obstacles). If the reflex controller suddenly halts, the robot will come to rest at the last computed safe subgoal. Such a failure is logical rather than physical: the robot fails to make progress.

An important feature of reflex control is that, when properly implemented, it does not interfere with execution of time-optimal control unless necessary. High-speed dynamics falls within the regime of competence of the sliding-mode controller. In simple environments, i.e., if obstacles do not block the default time-optimal trajectory, the reflex controller permits the sliding-mode controller to perform the task which it does well. Only when the complexity of the environment is beyond the reasoning of the sliding-mode controller, i.e., an obstacle blocking the default path, is the presence of the reflex controller apparent.

Generalizing from the above progression, the next level of control should be somewhat more abstract, but not monumentally so. The problem domain and solution approach should be restricted to a level in which decisions can be made within an appropriate time scale. Decision rates which are faster than reflex subgoal generation rates are permitted, though the reflex controller would not make use of the faster decisions. The excess computation capacity could be put to better use by absorbing more complexity in the higher level task. Slower decisions are permissible, since the reflexes would continue to keep the robot safe. However, below some time scale defined by the problem domain, too slow of a decision rate will result in poor performance or failure of the controller.

Failure at this level is more abstract than the obvious consequences of failure to control current, failure to control mechanical dynamics, or failure to prevent collisions. A planning-level failure manifests itself as a failure to progress towards a goal. In dynamic environments, this type of failure would be common if a global planner were used directly in conjunction with reflexes. The relatively slow rate of solution updates (typically orders of magnitude slower than reflex subgoal updates) would result in computed paths which were invalid before they could be executed. At least one intermediate level of planning is required to bridge the gap between global planning and reflexes.

Generalizing further from the preceding, it is desirable that the next level of control above reflexes should not interfere with reflex control or with sliding-mode control when either of these levels is competent in handling the complexity of the environment. Higher levels should be transparent when no obstacles are present, thus permitting the sliding-mode controller to execute time-optimal-motion. Further, in environments containing simple obstructions, reflex control should be permitted to exert its influence without interference from higher levels. If a minor deflection from default optimal control is all that is required, then reflex control is more effective than a path planner, which is incompetent at considering dynamics. In this regime, reflex control is expert; neither higher nor lower levels could accomplish the coordination of high-speed motion and obstacle avoidance as effectively. However, more complex environments, e.g. mazes, are beyond the capability of the reflexes, (even for trivial mazes).

Finally, it would be desirable for higher levels of control to satisfy the fail-safe property exhibited by the lower level controllers. That is, failures at higher levels should not incapacitate the lower levels.

From the preceding observations, four basic properties of an integrated control scheme are conjectured:

1. successive layers of control should be competent in incrementally more abstract problem domains;

2. successive layers of control are permitted successively longer cycle execution times; a minimum time for competence within the respective problem domain is determined by the dynamics of the problem at the respective level.

3. higher levels of control should fail safely; failure of any one level should not impair the competence of lower levels.

4. each layer of control has a regime of competence in which it is expert; no other layers of control should interfere with the expert level when the problem complexity falls within the expert's domain.

Of the above, the first point, organizing layers by levels of abstraction, is traditional. The important message of 1) is that the levels should be incrementally more complex. Attempts to tie high-level reasoning directly to low-level servo control would be ineffective. Intermediate levels of increasing abstraction should be included. Combination of incremental levels of abstraction has been explored and recommended by researchers at the National Bureau of Standards [7]. In the NBS system, many layers of control are combined, where each layer is constrained by a design guideline to consider no more than seven events. Separate processors at each level are implemented as finite state machines, all of which are synchronized. Control by synchronous state machines guarantees that the speed of response at each level is sufficiently fast. In fact, the response of higher levels is unnecessarily fast, since 2) suggests that higher levels are permitted longer time constants. The harsh restrictions on complexity (seven items) and implementation (decision tables) artificially excludes the use of algorithms, functions, and deductive logic. Further, the NBS approach violates propositions 3) and 4). The structure is dogmatically authoritarian, and does not delegate responsibility to levels of appropriate expertise. All commands and corrections come from above, and lower levels are not competent in the absence of higher levels.

Reflex control would not be consistent with the NBS architecture. Reflexes, as presented here, are capable of competent goal acquisition in simple environments, without the existence of separate levels of joint coordination and trajectory planning. More importantly, reflexes have the capacity to override orders from above. The NBS approach is orderly, but sacrifices efficiency and robustness by failing to accommodate points 2) through 4).

A layered control structure approach which is more similar to the present proposal is described by Brooks in [10,11]. Brooks defines control layers which are "subsumed" by additional layers with increasing levels of competence. In this work, a mobile robot control system is described in which 3 levels of control have been defined: level 0, in which the robot stops before obstacles or backs off from approaching obstacles: level 1, in which "wandering" is introduced in conjunction with obstacle avoidance; and level 2, which imposes an "exploratory" behavior on the wandering.

Brooks' layered approach, called a "subsumption architecture", roughly conforms with the suggested guidelines 1) through 4). Successive layers have increasing levels of competence, longer time cycles are exhibited by successively higher layers, and lower levels do not depend on higher levels for proper execution. In contrast, though, subsumption architecture as implemented has assumed very coarse steps in levels of abstraction and in cycle times. Even at level 1, new headings are only generated every 10 seconds. More importantly, the subsumption architecture does not recognize levels of expertise; it is always assumed that the highest active layer is the most capable control level. Only higher levels are permitted to suppress lower levels, not vice versa. No high-speed controller has been implemented, but were one to exist, it would be defeated by the presence of a higher-level planner. The subsumption architecture is intended to allow easy interface with higher levels. However, the addition of higher levels does not necessarily preserve the contribution of lower levels. In the instantiation, level 2 is implemented by suppressing level 1. The "wander" layer has a level of competence when it is not subsumed, but it does not seem to contribute to the system performance after the addition of level 2.

Hardware implementation of the system described here is closer to the NBS realization than Brooks's realization. The controller presented here uses multiple processors on a common bus with communication via shared memory, as in the NBS system. The subsumption architecture uses no common bus and no shared memory. On the other hand, the NBS system synchronizes all processes, whereas the present controller runs completely asynchronous programs (except for chip-level bus arbitration), as does subsumption.

In conforming to the suggested control integration guidelines, a control level above reflex control is sought which is incrementally more competent, may have somewhat slower cycle times, and should preserve the desirable features of the lower levels. Since local planning utilizes only local information, it is closely related to reflexes, and is thus a prime candidate for the next level of control. Local planning may augment reflexes by specifying goals to the reflexes. The reflexes in turn would break down the input goals into reflex subgoals. Whenever a goal specification is within the reflexes' lookahead window and is not occluded by an obstacle, then the reflex subgoal would be identical to the local planner's goal. Thus, sequential subgoals delivered by a local planner can lead the robot, via the reflexes, through a path which is not restricted to a simple potential function description. On the other hand, if the planning-level goal which is delivered to the reflex controller coincides with the ultimate target location, then the local planner will be transparent. Incremental goals generated by the local planner may only be introduced when it has been determined that the problem complexity is beyond the regime of competence of the lower levels. This is the approach which has been implemented. In the next section, the particular local planning technique used is described.

§4. Variations on Local Planning

In this section, variations on a local planning technique will be detailed which adapt the method for integration with reflex control and global planning in dynamic environments. It is not suggested that the chosen technique is optimal or general; it serves primarily to illustrate integration concepts. The crucial aspect of the selected algorithm is that it utilizes only local information. Alternative local planning techniques may be employed in the manner presented here.

A simple local planning algorithm (and extensions) is described by V. Lumelsky in [28,29,30]. Lumelsky refers to the algorithm and its variations as "Bug" algorithms, since the problem statement resembles that of a small automaton moving on the surface of a 2-dimensional environment with obstacles. The "bug" is aware of its current position and the position of its goal, but it does not know the terrain between itself and the goal. Obstacles are perceived only locally, e.g._ by "feeling" contact with an obstacle boundary. The algorithm invokes the use of a "line of preferred motion" referred to as the "M-line", which is a default path connecting the start location, S, and the target location, T. When the automaton, while moving along the M-line, encounters an obstacle boundary, it defines the contact location as a "hit point", labeled H_j. At the obstacle boundary, the automaton follows the contour of the obstacle (either clockwise or counterclockwise) until it once again resumes motion along the M-line. Upon leaving the edge of an obstacle to follow the M-line, a "leave point", L_j, is defined. The algorithm consists of two steps:

1. From point L_{j-1}, move along the M-line until: a) the target is reached, or b) an obstacle is encountered, and a hit point M_j is defined. Go to step 2.

2. Follow the obstacle boundary until: a) the target is reached; or b) the M-line is met at a distance $d < d(H_j, T)$, then define leave point L_j, increment j and go to 1; or c) the automaton returns to H_j without ever meeting the M-line. This indicates that the algorithm is not converging on the goal.

The algorithm is illustrated in figure 1. The "bug" starts at the point labeled "S" and initially follows the M-line, a straight line from "S" to "T", the target point. At point "H" an obstacle wall is encountered and a hit point is defined. The wall is followed (counter- clockwise in this case) until the M-line is encountered at point "L", which is closer to the target point than the last hit point, "H". A leave point is then defined at "L", and the bug continues by following the M-line to the target point.

In a plane, case 2c) can only occur when there is no feasible path to the goal. If a solution exists, then the algorithm is guaranteed to converge on a solution. If the two-dimensional environment is the surface of a torus, then case 2c) may occur although a valid solution exists. However, a new M-line must be chosen to find the solution. The difference between the two environments is that on a plane, there is a unique straight-line vector from S to T, whereas on a torus there are four distinct paths from S to T which are linear in the space variables and which-monotonically approach the target (approach from plus or minus θ_1 and plus or minus θ_2). It can be shown that it is only necessary to repeat the algorithm for at most two of the four M-lines to either find a solution or determine conclusively that none exists [30].

The preceding algorithm was proposed as a solution technique for two dimensional static mazes where extremely little information is known about the environment. It has the advantages that it does not depend on complete information, and it is guaranteed to converge on a solution whenever a solution exists. It has the disadvantage that the resulting solution is typically much less efficient than a globally optimized path. The solution process may require an exhaustive search of all obstacle boundaries, and the search process frequently involves retracing former path segments. Such properties, however, are inherent in local techniques. Without global information, optimal paths can not be deduced, and without the default of an exhaustive search, convergence (when a solution exists) can not be guaranteed.

A more serious limitation of the algorithm is that it is not generalizable to higher dimensions. In two dimensions, the algorithm works by effectively restricting the search space to a subset of candidate paths which are known to lie in the set of possible solutions. The candidate paths are those which consist of obstacle borders and segments of the M-line. The start, target, hit and leave points are equivalent to nodes in a graph of all candidate paths, where the nodes are connected by edges of obstacles or by segments of the M-line. For problems in 2-dimensions, the graph has a countable number of nodes and branches. Although the local planner does not have a priori knowledge of the graph structure, its algorithm is equivalent to an exhaustive search of the graph. In higher dimensions, the obstacle borders are not simple line segments, but are surfaces or hypersurfaces. Thus, the options for choosing a direction of motion along the obstacle wall are not limited to "right" or "left"; an infinite selection of directions is possible along the surface. Thus, the graph of path options is not finite, and it can not be searched exhaustively.

As an illustration of the difficulty of local planning in higher dimensions, consider the following scenario. A point automaton in 3-dimensional space is separated from its goal by an obstacle which corresponds to a spherical shell which encloses the goal. If the spherical shell is punctured at a single point, then feasible paths to the goal exist, where all feasible paths must pass through the puncture. Without knowledge of the existence and location of the puncture, any local technique which guarantees convergence will degenerate to an attempt to exhaustively search the surface of the punctured sphere. Thus, restriction of the local planning problem to the paucity of information proposed in [28,29,30] would result in intolerably inefficient algorithms in higher dimensions. Some additional information about the environment is required in order to construct useful planning algorithms, although full use of complete information should not be necessary.

For the two-dimensional planar manipulator described in Chapter II, the "bug" algorithm exhibits the type of properties desired in a level of planning which is intermediate between reflexes and global planning. In the present use of local planning, the motivation is not that of incomplete information: obstacles are introduced only at globally sensed or programmed locations. Complete information of obstacle contours is represented in discretized configuration space. Instead, local planning is employed because, by virtue of using only local information, extremely fast decisions can be made, albeit not as sophisticated as a global planner. The introduction of a control layer which is incrementally more sophisticated in its use of logic is consistent with the prior arguments.

Noticeable improvement in the efficiency of the "Bug" algorithms can be obtained by relaxing the assumed restrictions somewhat, while preserving the motivation of the approach. The premise of limited information is modified here include a "tunnel vision" capability. That is, it is assumed that the automaton is capable of examining a line of sight in the direction of the goal. If the goal is "seen" unobstructed from any location, then the automaton is permitted to abort the planning algorithm and head straight for the goal. Although this assumption utilizes more than strictly local information, the extra information employed is quite restricted, and its use does not increase the level of complexity substantially. Inspection of a straight-line path in configuration space involves a relatively small number of pixel examinations. Further, the line-of-sight search is no more complex in higher dimensions; the number of pixels involved is only a function of the distance between the robot and its goal.

The utility of this variation is illustrated in figure 1. In this example, a single non-convex obstacle blocks the robot's path from the start position (S) to the target (T). In the bug algorithm, an M-line is constructed from S to T, which defines a hit point, H, and a leave point, L. Following the bug algorithm, the automaton would proceed from S along the M-line to H, then circulate about the obstacle in the chosen direction (in this case, counterclockwise, or a positive circulation). Boundary following would continue until the point L is reached, then motion along the M-line would be resumed. Although the algorithm is successful, the resultant solution is highly inefficient. A seemingly natural improvement is to

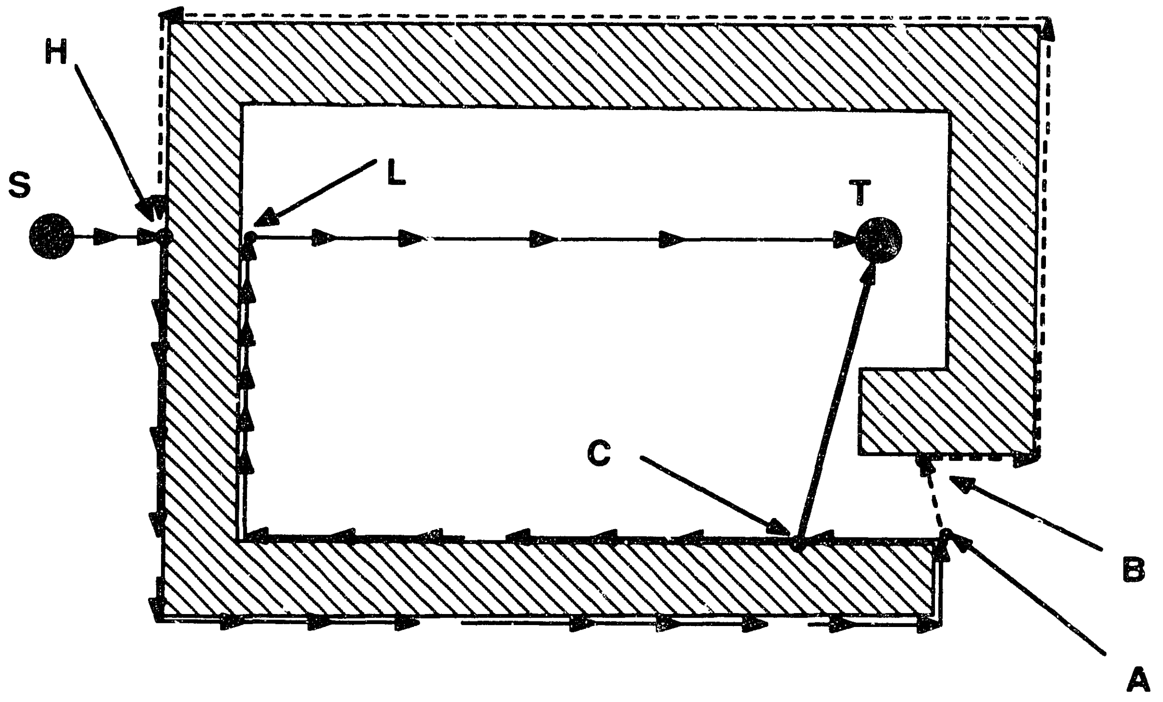

Figure 1: Bug Algorithm and Variations

permit unconstrained motion toward the goal as soon as forward progress is not blocked by an obstacle. The failing of such an approach is also illustrated in figure 1. Initially, the automaton would follow the same path as the bug algorithm, but when it reached point A it would break away from the wall in an attempt to reach the goal directly. However, at point B it would encounter another barrier, and resort back to wall following. Continued wall following would bring the automaton back to point H, after which the cycle would repeat in an infinite loop.

With the proposed line-of-sight inspection variation, the automaton would follow the bug algorithm up to point C, at which time a clear line-of-sight path to the goal is observed. The automaton would then break away from the wall and successfully travel straight to the goal. The resultant path is, in the present example, substantially shorter than that of the original bug algorithm. Convergence guarantees of the bug algorithm are preserved. The line-of-sight variation can improve on the bug algorithm, but will never result in a less efficient solution.

An additional variation on the line of sight and M-line constructions adapts the bug algorithm for effective integration with lower control levels. That is, motion along the M-line or along the goal line-of-sight is not constrained to follow a straight line, but is instead permitted to execute time-optimal control toward the goal. In proving convergence of the bug algorithm, it was not necessary to restrict the M-line to straight lines. Any non-looping path which connects the start and goal would satisfy the requirements. A particularly convenient choice for the main-path is the path which would result from executing time-optimal control towards the goal. In fact, it is not even necessary to define a static M-path; the purpose of the M-path is to define a valid leave-point for an obstacle boundary once wall-following has commenced, so that infinite loops will be prevented. A more general process for constructing the M-path is as follows: define the M-path as a sequence of path segments consisting of: 1) a path from the start location to the first point of an obstacle wall (or to the goal, if no obstacle is encountered) at which time-optimal control towards the goal, in combination with reflex control, would result in stalled motion. This point is unique in any static environment, and the resulting path approaches the goal monotonically. It will be referred to as a stall point. This path connects with: 2) a straight line-segment from the stall point towards the goal, terminating in a leave-point, as in the bug algorithm. To complete the path, repeat definitions 1) and 2) until the goal is reached. The prescribed path converges monotonically on the goal, and thus satisfies the requirements for a generalized M-line.

Computation of the preceding M-path may seem burdensome. In fact, computation of the M-path is not necessary; it is merely necessary to postulate the existence of such a path, and to conform to it when not following obstacle contours. For the prescribed M-path, conforming to the path corresponds to executing reflex and optimal control. These levels define the M-path, and thus perfectly execute motion along the M-path. It is unnecessary to anticipate exactly what that path will be, since following it is all that is required.

With this variation, the new algorithm is executed as follows:

1. from the start point, and from each leave point, L_{j-1}, assign the target position as the next goal to the reflex controller. If the reflexes result in stalled motion at an obstacle boundary, define a hit point, H_j. Go to 2.

2. Define the next segment of the M-path as a subset of the line segment from H_j to T. Assign a local planning level goal to the reflex controller which is the next pixel in configuration space which corresponds to positive rotation about the obstacle boundary (i.e., turn right). Continue assigning local-planning subgoals until a)the goal is reached, or b) the line segment connecting H_j and point T is encountered; define a leave point L_j and go to 1; or c) the automaton returns to point H_j without finding a leave point; select a new approach to the goal (i.e., reverse the angular direction of approach for both links) and start over.

In the preceding algorithm, the spirit and convergence properties of the bug algorithm are preserved, but use is made of the levels of expertise of the reflexes and optimal controller. Further efficiency improvements can be made by incorporating the line-of-sight search. However, proper implementation of this addition requires some variation. If a line of sight to the goal determines that wall-following may be aborted, then the line-of-sight path to the goal must be followed. Simply reverting to reflex and optimal control would not guarantee convergence. This is illustrated in figure 2. In this example, it is assumed that controlled motion is faster in the θ_2 direction than in the θ_1 direction (which is the case for the experimental planar robot). From point S, the robot heads toward the goal under optimal control. Near point H, however, the reflexes bring the robot to a halt. Further progress toward the goal is not possible under reflex control alone; the H point is equivalent to a local energy minimum. The local planning algorithm then steps in to initiate boundary following in a counterclockwise circulation. At point L, a clear line of sight exists to the goal. If the robot is permitted to break away from the wall at this point by assigning the target point as the next local planning subgoal, then optimal control will take over, driving the robot towards

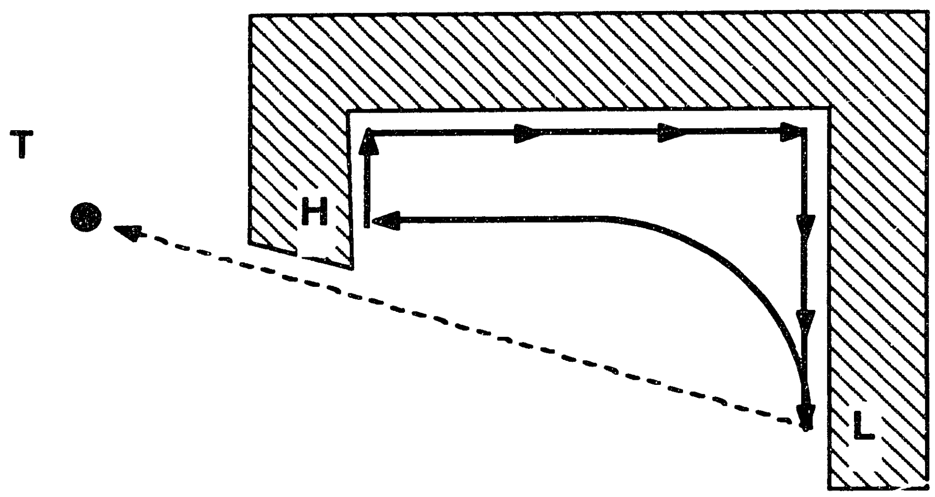

Figure 2: Line-of-Sight Looping

point T. However, optimal control does not happen to coincide with the line of sight path from L to T, and the robot ends up returning to point H, after which it will loop forever.

The efficiency of the line-of-sight variation may be recovered by at least two means. First, a breakaway initiated by a clear line-of-sight condition may impose the constraint of path following along the line of sight, as opposed to unrestrained high-speed servoing to the goal. A second, more efficient approach is to inspect for the goal not along a straight line of sight, but along the anticipated trajectory of postulated optimal control to the goal. Such a search corresponds to a simulation of the robot motion under optimal control. If the simulation indicates no collisions, then the robot may be released to execute high-speed motion to the goal. Since the algorithm described is particularly simple, the processor executing it would be unnecessarily fast. By guideline 2) of the preceding section, more problem complexity may be incorporated at this level in order to take advantage of the longer cycle time permitted. Thus, it would be consistent to incorporate the proposed trajectory simulation within the local planning level.

With the above variations, the bug algorithm is compatible with reflex control and optimal control. Optimal control is used for rapid transitions along the M-path segments, and reflex control continues to prevent obstacle collisions (at least for static obstacles). The reflexes are also still active during wall-following. Normally, local planning subgoals would coincide with reflex subgoals, and the reflex controller would be transparent to the local planner. However, if an obstacle moves between local planner subgoals, or if the local planner fails completely, the reflex controller will continue to assign safe subgoals to the robot. Moving obstacles, however, present additional complications for the local planner.

As discussed in Chapter V, it is impossible to guarantee prevention of collisions in dynamic environments, even if the obstacles are moving arbitrarily slowly. Analogously, in dynamic environments, it is impossible for any planning technique to guarantee successful guidance of an automaton to a goal, although a solution may exist. At any crucial juncture in the algorithm, malicious obstacles may surround the automaton and force it back to its starting point. Alternatively, a moving goal may choose to hide itself from the automaton by remaining in the "shadow" of an obstacle. As the automaton moves, the goal may also move to keep an obstacle between them (e.g., as though the automaton were "chasing" the goal around a circular table). The consequence of this statement is that it is fruitless to search for general proofs of convergence for planning in dynamic environments. This does not mean, however, that all planners are equally (in-)effective in dynamic environments. The bug algorithm can be made to be reasonably adaptive to changing conditions.

Since reflex control runs in parallel with local planning, unexpected motion of an obstacle boundary during wall-following can be safely accommodated. Sudden obstacle motion is actually more realistic than it might seem, since sensor noise results in an apparent motion of the obstacles. The perceived motion is indistinguishable from actual motion. Thus, procedures for accommodating unexpected obstacle motion (whether real or perceived) is required in any practical system. If an obstacle wall moves into the robot, the reflexes will force the robot to back away. On the other hand, if the wall suddenly recedes from the automaton, then the rule for circulating counter-clockwise about the obstacle will be invalidated. Simply reverting to reflex control at this point can result in losing the progress obtained by earlier wall-following. Sensor noise could have the effect of repeatedly dislodging the automaton from footholds while it attempts to scale a potential energy incline. Rather than reverting to reflex control when a wall is lost, the modified local planner begins to head in the direction in which the wall was last felt. If the wall is found, contour following is resumed. If the wall is not rediscovered after a brief search, then the target point is assigned as the next local-planning subgoal, and control thus reverts to the reflexes.

Another difficulty with moving obstacle boundaries is that the current hit point, which is used to detect cycling, may no longer lie adjacent to the obstacle wall if the wall moves. When the obstacle wall moves, the hit point may lie in free space or lie in the interior of the obstacle. In either event, under continued wall following the automaton will fail to revisit the hit point, and the test for cycling will never be satisfied. Since sensor noise will cause the apparent obstacle boundaries to fluctuate, in practice the hit point will rarely be revisited. A minor variation will reduce this sensitivity: on each cycle of the local planning loop, the hit point may be reassigned to lie adjacent to the nearest obstacle wall along the M-line. The effect is to make the hit point track the wall, as though glued to it. Only a search along a line is required, which may be limited to several pixels, so this variation does not consume appreciable computing time.

Tests for the leave point are already robust with respect to moving obstacle boundaries, since it is only necessary to determine whether the system has crossed the M-line while wall-following to detect a candidate leave point. This test remains valid, although the would-be leave point may vary as the obstacle moves. An additional requirement of a leave point is that the distance from a candidate leave point to the goal must be shorter than the distance from the last hit point to the goal. If the hit point is made to track the obstacle wall, then the distance test will remain valid (the test may be invalid otherwise). These variations will make the bug algorithm practical in the presence of sensor noise.

Of course, the resulting algorithm is not foolproof; no algorithm is guaranteed effective in the presence of moving obstacles. In the present case, the hit point may be surrounded by obstacles which move in after the automaton has left. Although the hit point remains on the surface of the original obstacle, that surface may no longer be reachable by the robot, in which case the test for cycling will never be satisfied.

Further complications are introduced by moving goals. Although convergence can not be proven in general when the goal is moving, a desirable characteristic of an algorithm is that it should degenerate to a guaranteed convergence algorithm if the goal stops moving. For the present algorithm, if the robot is not wall-following (i.e., it is on the effective M-path), then it will operate under optimal control servoing to the moving target. In effect, the M-line would be tracking the target, which invalidates the convergence proof. If, however, the obstacle should stop moving before the automaton has encountered a hit point, then continued-application of the modified bug algorithm will guarantee convergence to a solution (if a solution exists). If the goal moves during wall-following, the same claim can not be made. It may be necessary to revisit the original hit point in order to achieve a valid solution. If the solution process is restarted when the target stops moving, then the problem degenerates to the original case of guaranteed convergence in static environments.

Since it would be necessary to restart the solution if the goal moves during wall following, it is tempting to continuously redefine the M-path as a straight line from the current position to the goal location. Such an approach is not robust to minor perturbations, though, e.g. from sensor noise. If the goal is apparently continuously moving by a small amount, then continuous redefinition of the M-line can result in cycling, as illustrated in figure 1. On the other hand, the original M-line may be frozen, despite a moving goal, until a leave point is picked up. Such an approach is not at all responsive, though, to large motions of the goal. A proposed variation is to define a moving M-line between the hit point (which is "glued" to the obstacle wall) and the moving target. For small perturbations, this construction of the M-line converges on the static algorithm, but for large changes in the target position, the M-line responds by placing the leave point at a potentially more effective location. The proposed variation is more responsive to moving goals without being unnecessarily sensitive to noise. Still, any motion of the target during wall following invalidates former convergence proofs. Since a feasible solution may require revisiting the last hit point, the algorithm does not have a conclusive test for nonexistence of a solution whenever the target (or obstacles) moves. Consequently, the algorithm should continue to search for the goal, regardless of apparent cycling, until instructed to do otherwise by some higher level.

Most of the preceding modifications of the "bug" local planning algorithm have been implemented in an experimental robot control system. Moving goals and moving obstacles are accommodated by integrating sliding-mode time-optimal control, reflex control, and interactive local planning. The local planner follows obstacle boundaries by defining sequential goals for the reflex controller. Wall-following continues until: 1) a leave point is encountered, as defined by a moving M-line which pivots about the most recent hit point; 2) the obstacle wall is lost and cannot be rediscovered within a local search; or 3) an unobstructed line-of-sight path exists to the target. When any of these conditions occur, the local planner assigns the actual target position (which may be moving) to the reflex controller as a goal. The reflex controller then permits optimal control motion to the goal, or safely brings the robot to rest at a new hit point, upon which wall-following is reinitiated.

Improvements which have not been implemented to this point include updating the hit point to track moving obstacle walls, and simulating the trajectory of the robot in performing repeated tests for an unobstructed path to the goal. As a result, the system does exhibit cycling in situations where the hit point is lost, and in situations like that of figure 2.

A complete description of the experimental implementation of integrated local planning, reflexes and optimal control is given in the next section.

§5. Implementation of Integrated Control

An experimental realization of the proposed integrated, layered control system was implemented on a multiprocessor system. The computing environment, described in more detail in Chapter II, consisted of five MC68020-based single-board computers running on a common VME bus backplane. A separate memory board with 8 Mbyte of random access memory was installed in the backplane, and was available to all five processors as shared memory. Except for chip-level bus arbitration, the processors and their respective programs ran completely asynchronously. Communications among processors was via shared memory. Each processor board executed a single task, as described individually in Chapters II through VI.

The first processor was dedicated to acquiring sensory data regarding the positions of various obstacles, as well as a (possibly moving) goal location. As described in Chapter II, this board sequentially flashed infrared LED's which marked the locations of the obstacles as well as the goal. Analog optical position data was converted to digital form and digitally filtered. Resulting positions were continuously updated in a table in global memory at a rate of 500Hz per obstacle. Eight separate sets of LED's were sensed and recorded. A lookup table for interpretation of the position table values was also stored in global memory.

A second processor was responsible for executing the algorithms described in Chapter IV for fast configuration space transforms. This processor continuously cycled through the positions of each of the obstacles, as indicated by the most recent values in the sensor position table, and computed the corresponding forbidden joint angles in configuration space. A 128x128 discretization of 2-D configuration space was continuously updated in shared memory by the transform processor. Gray scale values of edge pixels were used to account for overlapping obstacles; edge normal information was also stored. Computed or hand taught static obstacles were also overlayed on the dynamically computed obstacle borders. Configuration space transforms were executed at (nominally) 50Hz. A means for efficient and robust parallelization of configuration space transform computations was described in Chapter IV, although this was not implemented in the experimental system. A single processor for map updates was found adequate in 2-D.

The next processor performed sliding-mode optimal control of the 2-D planar robot, as described in Chapter III. Both axes were controlled by the same processor, as well as both observers for velocity estimates. Custom electronics for sensing the robot link positions and for digital to analog conversion of PWM amplifier current commands were interfaced to this processor via a private bus. No VME bus accesses were required for robot communications. The sliding-mode controllers and the observers executed at 750 Hz. On each cycle, the controllers sampled locations in shared memory which contained goals specified for the sliding-mode controller. Actually, the goal position for the servo level consisted of a sequence of subgoal positions, as approved by the reflex controller. The servo board also returned information to global memory: the current robot position (joint angles); the current angular velocities (as estimated by the observers); and the current arrest point (the closest point at which the robot could be brought to a complete halt upon immediate maximum braking).

The fourth processor supported the execution of reflex control, as described in Chapter V. The local planner continuously polled the robot position and arrest point, and continuously examined a local region (limited lookahead window) in the configuration space map to determine whether obstacle evasion was necessary. The reflexes examined locations in shared memory in which the local planner specified "request" points. The request points were considered by the reflexes and, if it was determined that a request point was contained within the lookahead window in configuration space and was guaranteed safe, then the request point was approved as a set point and delivered to the sliding-mode controller as a goal.

Alternatively, unsafe requests or requests outside the lookahead window resulted in reflex subgoals which, with respect to the requested position, corresponded to the closest safe position within the lookahead window. The reflex controller also monitored the robot position with respect to configuration space to determine if obstacles were approaching. Escape from an approaching obstacle was given a higher subgoal priority than advancement toward a requested position. The reflex subgoal generation rate varied from about 600 Hz to about 200 Hz, (where the fastest execution occurred in the most cluttered environments).

The last processor employed executed the local planning algorithm described in this chapter, including the variations for control integration and dynamic environments. The local planner switched between two operating modes: free mode and wall-following mode. In free mode, the target position was read from the position table in global memory (as computed by the sensory processing board), and redeposited in shared memory at the drop point for request positions considered by the reflexes. In free mode, the local planner thus delegated full control authority to the reflexes and sliding mode controllers. While in free mode, the reflex controller continuously tested for a stall condition, which indicated that the problem domain was too difficult for reflex control alone. Stall conditions occurred when both links were blocked from further progress, or when one of the links reached its goal, but the other was blocked. Upon observing a stall condition, the local planner switched to wall-follow mode. In this mode, successive points along the boundary of the current blocking obstacle were delivered to the reflex controller as requested positions. Since the reflexes and local planner used the same configuration space map, all local planning wall-following requests were generally approved by the reflexes, and passed to the sliding mode servos. In addition to providing incremental wall-following requests, the local planner also continuously scanned for a clear line-of-site path to the goal in configuration space. Wall-following mode continued until: 1) a leave point was detected; or 2) a clear path to the goal was discovered, or 3) the current obstacle wall was lost. Upon leaving wall-following mode the local planner resumed free mode, and

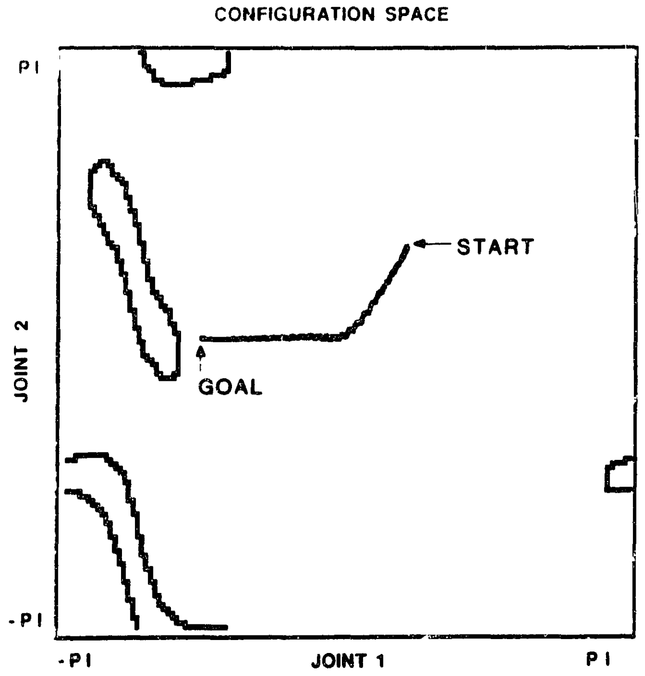

Figure 3: Robot Transient; No Reflex or Planning Influence

delivered actual target positions as request points. The local planner executed its cycles at a rate of between 500 Hz and 1kHz, which is unnecessarily fast.

Examples of the three controllers running in coordination are shown in figures 3, 4 and 5. The figures are actual recordings of the robot's motion in a cluttered environment, not simulation. In the first case, figure 3, the goal is not blocked by an obstacle. The reflexes and local planner do not interfere with the sliding mode controller, and high-speed target acquisition is accomplished. In the next example, figure 4, optimal control alone is not adequate to achieve safe goal acquisition. The reflexes step in to divert the robot's motion away from an impending collision. The resulting diversion is adequate in itself to achieve a valid path to the goal; no interference from the local planner is necessary, and none is introduced. As the robot clears the obstacle, optimal control is resumed.

In the third example, the robot is unsuccessful in reaching its goal under potential function (reflex) control alone. Help from the local planner is required.

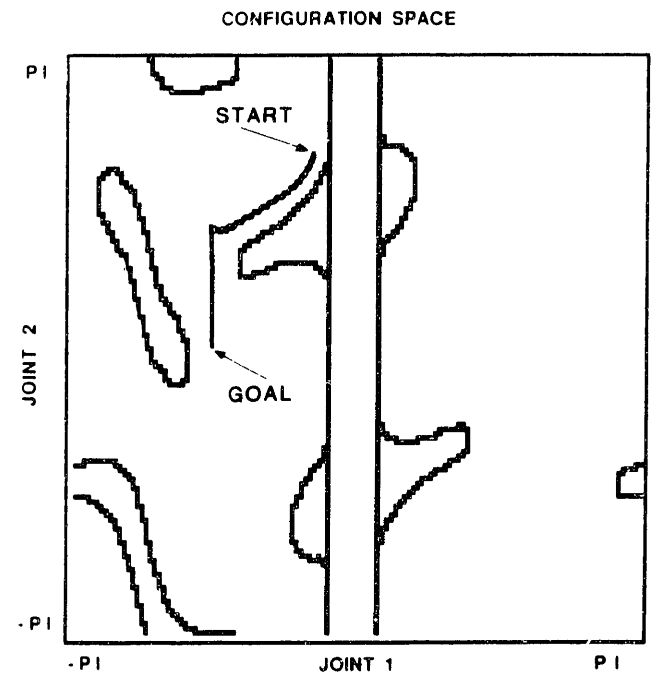

Figure 4: Robot Transient With Reflexes; No Planning Influence

Wall-following is introduced when the reflexes fail to make progress. As the perimeter of the obstacle is followed, the local planner eventually observes a line-of-sight clear path to the goal. At this point, it relinquishes control to the reflexes by assigning the actual target location as the next subgoal. The reflexes in turn permit the optimal controller to drive the robot, which is accomplished by delivering a rapid sequence of reflex subgoals at the edge of successive reflex lookahead windows.

The implementation exhibits the desired combination of capabilities: high-speed motion, obstacle avoidance, and (primitive) intelligence are all demonstrated without compromising system performance.

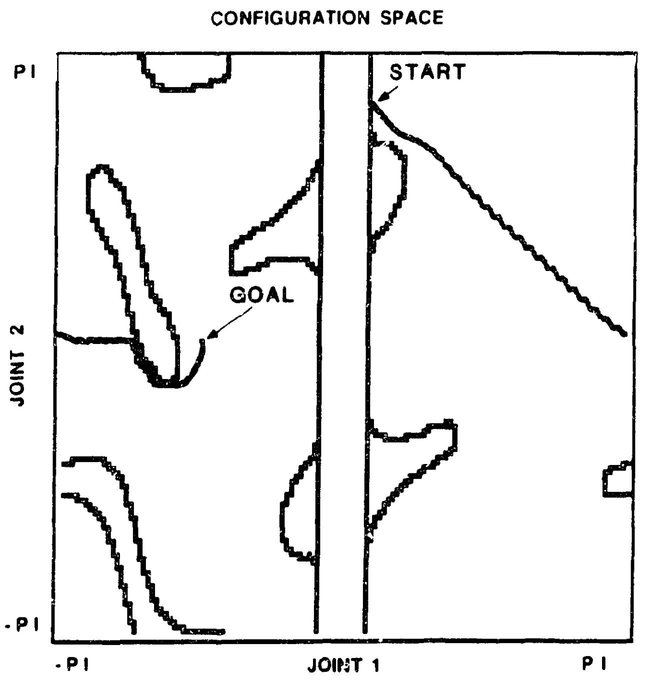

Figure 5: Robot Transient With Reflexes and Local Planning

§6. Summary and Conclusions

Effective integration of optimal control, obstacle avoidance and low-level planning has been achieved through a layered control approach. Design integration is characterized by several properties: each layer is incrementally more abstract; each layer is permitted incrementally longer cycle times; each layer builds on lower levels, and does not affect the performance of deeper levels; each level has a regime in which it is expert, and in which it may assume responsibility.

The control organization is distinct from alternative approaches particularly in the recognition of levels of expertise. Lower levels of control are not merely subservient to higher levels; each level covers a regime in which it is better able to perform a task than any other level. Thus, it is important that higher levels recognize their own limitations, and grant full responsibility to the most competent module at any instant. In this manner, system performance is not impaired by the existence of additional layers of control.

In extending the control system into the planning regime, it was determined that an intermediate level was required between reflexes and a global path planner. A local planning algorithm was selected and implemented. The local planner executed its cycles at a loop rate which is much faster than would be possible with any global planner, which permits the robot to react immediately rather than stall when confronted with a confusing environment.

The local planner implementation resulted in an execution rate between 500 Hz and 1kHz. Such a high rate of execution is unnecessary, though not harmful. The implication is that the restrictions on the amount of information considered may be further relaxed. Perhaps some form of "myopic" rather than strictly blind or tunnel-vision assumption should be employed to broaden the problem domain. Further, some limited lookahead in a planning (strategic) sense could be employed, analogous to the limited lookahead used by the reflexes for dynamics. According to the suggested design guidelines, though, the context of local planning should be broadened to utilize the time permitted for this level. A jump to complete global planning, though, would be too extreme.

At the time of writing, no higher levels of control have been implemented. The next natural layer is some form of more sophisticated planning, which may be a complete globally optimizing planner, if the jump in abstraction and execution time is not too large. Following the philosophy of the preceding layers, the next layer should interpret an ultimate goal location in terms of subgoals, which in turn get passed to the local planner. The local planner may be generalized slightly to hunt for efficient paths between globally planned subgoals. Such an approach would permit the global planner to plan its solutions faster, since a coarser description of configuration space may be used. Effective execution of a crudely planned path could be performed by the local planner, reflexes, and optimal controller. Presumably, the inferences drawn for design guidelines of effective integrated control systems extends to higher levels still, though the demonstration of such a claim would require further investigation.

CHAPTER VII

CONCLUSION

This thesis has presented a broad, but by no means exhaustive overview of issues in high-performance robotics. A ground-up approach was taken, starting from fundamental issues in design and extending through investigations in planning in dynamic environments. Five major areas were involved: design of a high-speed robot arm; robust time-optimal control; fast computations of moving obstacles in configuration space; potential-function based obstacle avoidance; and primitive path planning. In each case, the problem domain was restricted by simplifying assumptions, though issues of generality were considered throughout. Further, the simplifications were realistic in the sense that results of the work are at least in part immediately applicable in practical industrial systems.

All developments were considered theoretically and tested in hardware. Effective performance in each of the areas treated was measured not only by the efficiency and competence of each respective module, but also by the ability of the modules to work in cooperation. Factors influencing the limits to robot performance were identified in each of the five a.eas, as well as in the structure of their integration.

In the area of design, limitations to speed are relatively clear: actuator effort and velocity limits, actuator electrical dynamics and link and actuator inertias impose physical limitations on the ultimate speed of a given machine. The use of lightweight link construction and overpowered, high bandwidth, remote actuators permitted very high accelerations of the experimental arm. Somewhat less obvious performance implications of design include the introduction of nonidealities which make the system difficult to control. Transmission friction and backlash, and system resonances limit the performance of the robot under feedback control to speeds well below the open-loop physical limits. In the design described here, traction drives (pretensioned steel bands) were used to obtain zero backlash, very low friction power transmission. In this manner, the virtues of direct-drive design were realized without the penalty of drastically mismatching load and actuator inertias. The concept of direct-drive actuation in the strict sense was rejected; directly connecting loads to actuator armatures is generally grossly suboptimal, and does not permit remote actuation. In spirit, though, the direct-drive philosophy was followed, in that a virtually ideal transmission was used.

One final critical design feature was the implementation of counterbalancing for dynamic decoupling. Decoupling provided such a dramatic simplification of the system equations that sophisticated controls, virtually impossible to realize on nonlinear coupled systems, became almost trivial to implement on the decoupled arm. At the same time, however, this simplification places some limits on the generality of the techniques and results presented here. Dynamic decoupling remains valid as long as the robot does not handle payloads which significantly alter the arm inertia tensor. Ideally, robots of the future would handle payloads substantially heavier than the arm itself. The influence and compensation of heavy payloads with respect to dynamic decoupling is an area of research which has not been treated here.

With careful attention to design, an arm with the physical capacity for high-speed controlled motion can be built. Conventional linear control techniques, however, are not adequate for utilizing the full capacity of a system. Linear control analysis presumes a linear system,which implies that actuator effort saturation must be avoided. In contrast, the theoretically best (fastest) possible control requires full actuator saturation at all times. Optimal control, however, is impractical since it is extremely sensitive to modeling errors and disturbances in open-loop form, and is generally unstable in closed-loop form. A robust form of nearly time-optimal nonlinear control was derived here theoretically and demonstrated on the high-speed arm. The approach invoked sliding mode control with smoothing, a technique with its roots in time-optimal control. Here, sliding-mode control has been re-united with time-optimal control to achieve robustness. Roughly, a factor of four in speed improvement was obtained over linear control.

An ability to perform fast, controlled motion of an arm is not sufficient in itself to achieve competent robot behavior. In fact, higher arm speeds accentuate the penalty for colliding with an obstacle. In the present approach, no path constraints were specified for the (near) time-optimal controller. Indeed, no path at all was specified for the controller, and thus no consideration of obstacle avoidance was involved. Instead, a second layer of control was built to enforce obstacle avoidance during high-speed motion. This level of control was labeled "reflex control", as its behavior was suggestive of reflexes. Within the reflex controller, rudimentary processing of local sensory data was performed and immediate action was taken if emergency conditions existed. Otherwise, the reflex controller was "transparent"; its influence was not exerted unless absolutely necessary. In this manner, default high-speed servoing was performed as though no obstacles were present. As long as the resulting trajectory (not a pre-computed trajectory) was safe from collisions, maximum speed was not inhibited. In the event of impending collision, though, the reflex controller exerted its influence to deflect the robot away from obstacles.

Reflex control was derived from variations on potential function based control. A new feature introduced was explicit consideration of actuator saturation in constructing obstacle repulsion fields for guaranteed feasible obstacle avoidance. Prior techniques guaranteed obstacle avoidance only mathematically, with the unrealistic assumption of infinite actuator capacity. A further addition was the definition of minimally influential protective potential fields. Such fields preserve feasibility and safety requirements while exerting repulsion over a minimum range. Finally efficiency of the control computations was dramatically improved through the definition of a minimum inspection window which contained the minimum set of all crucial obstacle avoidance data at any instant as a function of the state of the robot. All of these features were made possible through a simple departure from prior work in potential function control. Specifically, potential functions were combined using a logical operation (supremum operation) as opposed to an arithmetic operation (addition of functions).

In order to execute both reflex control and high-speed servoing efficiently, a common representation was required which was suitable for both control layers. In terms of time-optimal control, the choice of reference frame was clear: only the space spanned by absolute joint angles was suitable for high-speed control. Time-optimal control in any other frame would have been totally impractical. Similarly, explicit consideration of actuator saturation in obstacle avoidance also demanded a reference frame in which actuator effects were decoupled. In order to describe obstacles in terms of the preferred reference frame, it was necessary to compute transformations of task-space obstacles into configuration-space obstacles. In order to accomplish such transformations in real time, the transformation process was studied mathematically, and efficient algorithms were derived. A method for constructive geometry of configuration-space obstacles was proposed and implemented using primitives which are invariant with respect to rotations (points, circles, spheres). In two dimensions, simple moving obstacles were transformed to configuration space at video rates. A multiprocessing algorithm was proposed for implementation of fast configuration-space transforms in higher dimensions.

The levels of time-optimal control and reflex control were combined through the common representation of configuration space. Competence in high-speed motion was provided by servo-level control, and limited awareness and reaction to a sensed environment was included with reflex control. The resulting control system, though, was not sufficient to obtain effective autonomous behavior. In simple (relatively uncluttered) environments, nearly time-optimal motion was achieved. In more complex environments, though the potential-function based obstacle avoidance controller often resulted in stagnation at local energy minima. To achieve a higher level of competence, a third layer of control was added: local planning. An algorithm which only used local information (locations of nearby obstacle boundaries) was chosen as an intermediate level between fast reflexes and relatively slow globally optimizing planners. In all cases, configuration space provided a highly efficient common representation.

In integrating the three layers of control, a prime objective was the preservation of high speed performance as additional levels of complexity were incorporated. Certainly, arbitrarily complex behavior could be realized if arbitrarily long computation times are allowed, and if arbitrarily slowly evolving environments are considered. However, in such cases it would not be clear whether the approach would ever be compatible with high-performance systems. Instead, the chosen approach was to build up complexity incrementally without sacrificing speed at any level. In the process, attributes of efficient layered control systems were recognized, and control system design principles were inferred. They are: 1) successive layers of control should be competent in *incrementally* more abstract problem domains; 2) successive control layers are permitted successively longer-cycle execution times, as constrained by the dynamics within each problem domain; 3) higher levels of control should not impair the operation of lower levels of control, even upon total failure of the higher level; 4) each layer of control has a regime of competence in which it is expert, and in which no other layer should interfere.

A notable distinction of the above from traditional hierarchical control schemes is the recognition of regimes of expertise. A regime of expertise is not the same as a level of competence. Increasing levels of competence imply increasing levels of authority and capability. With distinct regimes of expertise, however, each layer of control assumes full authority when the problem description falls within its capabilities. No other layer of control, i.e. no "higher" level, should interfere with the operation of the defined expert in any particular situation. In the system described here, the servo level of control is given complete authority unless it is judged incompetent. The existence of a path planner does not imply the use of planning, or the decomposition of the goal acquisition problem into a sequence of simpler tasks. In many cases, servo control alone is adequate, indeed superior, in performing the task. Planning, which inevitably consumes time, should only be invoked when it is needed. Higher levels must recognize when their talents are required, and when to delegate authority. Conversely, lower levels may assume authority over higher levels, e.g. in reflexive response to sensed danger which overrides any logical plan prescribed by higher levels.

In the investigations described here, the proposed control system philosophy worked well. Broad generalizations to arbitrary systems are tempting, though this thesis does not pretend to prove or justify the control system guidelines in any rigorous sense. In part, the lure of broad generalization has been resisted by the author in recognition of the merits of specialization. While design and control techniques should be general, their proper use requires application-specific knowledge. In the present system, the implementation of high-speed control required a good dynamic model of the system. The reflex layer of control required knowledge of the performance of the servo controller in order to guarantee feasible obstacle avoidance. The planner required knowledge of the limitations of the reflex controller in order to evaluate when planning was necessary. The configuration space transforms required very specific knowledge of the robot's geometry and kinematics. In each case, high performance implementation required use of system-specific information.

The demonstrated effectiveness of fixed automation is convincing proof of the merits to design and control specialization. While the ultimate robot would be a highly adaptable machine, this does not imply that it would use a generic controller. An effective robot controller would, as in the present case, utilize a great deal of knowledge about the robot's own body and its own control system.

In each of the areas treated here, interesting unanswered questions and problems arose, which remain areas of future research. In the area of design, dynamic decoupling presented tremendous simplifications for time-optimal and reflex control. The reliance on dynamic decoupling immediately presents several crucial questions. How sensitive are the methods here to coupling through imbalance (e.g., by picking of a load of significant mass)? Can decoupling by design be extended to higher dimensions? (Certainly, a z-axis could be added for a decoupled SCARA-like design, but the wrist degrees of freedom would not be decoupled, and it has not been shown whether wrist dynamics have a negligible influence on the arm dynamics). How applicable are the methods to existing robot designs? Can dynamic decoupling be effectively imitated through the use of an inner feedback loop? Would such an approach compare favorably with feedback loops which make the robot imitate Cartesian dynamics?

The use of remote drive rather than direct drive emphasized the importance of transmissions in robot design. A nearly perfect transmission was implemented in the present case through the use of steel bands, and the virtues of direct drive were realized without the inertia penalties. The approach suggests the investigation of other types of traction drives in robot design. Further, transmission behavior might be improved through the use of force/torque sensing with local feedback around the transmission.

In the use of sliding-mode control for high-speed servoing, several avenues for additional research became apparent. In the implementation of near time-optimal control, a torque chatter was observed during deceleration. The controller was stable, as predicted by the Lyapunov proofs, but the chatter was unexpected. Use of saturation functions for torque smoothing should have eliminated the chatter. The source of chatter has not been rigorously identified. In addition to refining the current controller, extensions to higher-order systems should be possible. Robust time-optimal control of more complex systems, including some degree of dynamic coupling between links, should be investigated. More fundamentally, the adaptation of sliding-mode control for high-speed motion is just one example of applying sliding-mode theory for the realization of some specified set of dynamics. Dynamics other than that of optimal control could be specified, e.g. the dynamics specified to realize impedance control. A key point presented here is that sliding-mode theory has been used to control a system to realize a set of dynamics, not just a specified state or specified path.

In the area of fast configuration-space transformations, two basic ideas were presented, but not implemented. First, an approach for generating configuration space forbidden regions by means of a constructive geometry was described, but not implemented. Points, spheres and swept points and spheres were suggested as basic model elements and operations. Additional basic elements and operations may suggest themselves in future research. Implementation in 3-D would be a crucial test of the practicality of the techniques. A second major point which was not pursued is the suggestion of using swept or blurred obstacles in configuration space to account for anticipated obstacle motion or for uncertainty regarding possible obstacle motion. Avoidance of moving obstacles which may not be considered quasi-static will depend on incorporating anticipation of motion in configuration space.

In the implementation of fast configuration space transforms, a chief bottleneck in computation was the discretization of line segments into pixels. The application of specialized machine vision hardware may be useful in speeding up the algorithm. Speed improvements will be particularly important in higher dimensions.

A particularly important generalization of reflex control will be the avoidance of moving obstacles, especially other robots. In part, this work will require anticipation of obstacle motions in configuration space. A more difficult problem still is the coordination of multiple interacting machines. If multiple machines react to each other by trying to anticipate each other's path, then it is not clear whether they will interact stably. Explicit synchronization of multiple machines is undesirable, since it would require extensive programming, and would not be flexible or robust with respect to individual machine malfunctions. Ideally, multiple machines would interact effectively with a minimum amount of communication among them. How this might be implemented remains an open area of research.

The area of planning was treated here only superficially, in the context of layered control integration. For the local planner chosen, cycle execution rates were higher than necessary, suggesting that additional complexity should be incorporated at this level. The control integration guidelines suggest that at least one layer of planning should be implemented between reflex control and global planning, but the optimal choice of an intermediate layer is not specified. An important consideration in the design of a local planner is that it should integrate effectively with the next layer of control, presumably a global planner. A desirable attribute would be the ability to perform global planning in a space which is discretized more coarsely than the space used by the local planner, such that refinements are performed locally while global planning is performed more rapidly through the use of coarser approximations. To coordinate with the global planner, the local planner should introduce only perturbations on the global plan (whenever the global plan is valid), rather than attempt a significantly different approach.

The control integration design guidelines suggested here were implied by the results obtained. No rigorous proofs, deductions or rigorous generalizations have been offered. Of particular importance is the issue of stability in integration. Layers which are stable by themselves are not guaranteed to be stable in combination with additional layers of control. The suggested guidelines may have bearing on general conclusions about layered control system stability. In addition, the use of logical decision making in feedback control was required, but rigorous theory does not exist to suggest design rules or predict performance. Digital control theory does not apply to general decision making in control. Also, in the layered control system constructed, parallel processing was used. The approach was successful, but did not benefit from the guidance of any rigorous theory or design procedures for invoking parallel processing in control. These aspects, layered control theory, decision-making in control, and parallelism in control, should be rigorously formalized.

This thesis has presented some of the current limitations to high-performance robots, proposed some solutions, and demonstrated their implementation in a simple system. With proper design and control, robots are capable of dramatically better performance. Perhaps future robots will appear less human-like, but they may be expected to be more employable.

APPENDIX A

OBSERVER IMPLEMENTATION

In the design of the experimental apparatus, tachometers were included for velocity measurements. However, the tachometers were found to be too noisy at the low rpm's which resulted from controlled robot motion. Implementation of the sliding-mode time-optimal controller relied heavily on the actual link velocities. To obtain velocity estimates for feedback, observers were employed.

For each link, the system model may be described by the block diagram of figure 1. In figure 1, the state $\mathbf{x}$ is given by:

$$\mathbf{x} = \begin{bmatrix} \theta \\ \omega \end{bmatrix}$$

Ignoring friction, (which was known to be low) the system matrices are given by:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ \frac{1}{I} \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

For link 1, the value of I (link inertia) was 0.037 Kg-m^2 and for link 2, the inertia was 0.0035 Kg-m^2. Observers were constructed in software as illustrated in figure 2. In figure 2, the observer consists of a model of the plant with estimated values for the system matrices $\mathbf{B}$ and $\mathbf{A}$, $\hat{\mathbf{B}}$ and $\hat{\mathbf{A}}$ respectively. If the plant model used in the observer were perfect, and if the initial conditions were known perfectly, then the estimated state, $\tilde{\mathbf{x}}$, would equal the actual state, $\mathbf{x}$, for all time. For even minor errors in the estimates of the system matrices, however, the simulated state will eventually drift away from the actual system state. The observer is made to keep tracking the actual system by feeding back an error signal derived from the difference between the known (measured) components of the actual system and the corresponding estimated state variables. In the present case, the actual link positions were measured via optical encoders and compared with the estimated link positions. Assuming $\hat{\mathbf{A}} \approx \mathbf{A}$ and $\hat{\mathbf{B}} \approx \mathbf{B}$, the dynamics of the state estimate error, $\mathbf{e} = \tilde{\mathbf{x}} - \mathbf{x}$, may be expressed as:

$$\dot{\mathbf{e}} = \mathbf{A}\mathbf{e} - \mathbf{L}\mathbf{C}\mathbf{e}$$

The error dynamics has the characteristic equation given by:

$$s^2 + K_1 s + K_2 = 0$$

where K_1 and K_2 are defined by the observer feedback matrix, $\mathbf{L}$:

$$\mathbf{L} = \begin{bmatrix} K_1 \\ K_2 \end{bmatrix}$$

Since the sliding-mode controller was not designed to accommodate observer dynamics, the observer dynamics were made fast with respect to the controlled system dynamics. The value of K_2 was set to 400,000 (rad/sec)^2 to obtain error dynamics with a natural frequency of about 100 Hz. A damping factor of $\varsigma \approx 0.7$ was selected by setting K_1 to 900 rad/sec. Identical observer dynamics were chosen for both links.

Two factors limited the bandwidth of the observer error dynamics. First, the computation cycle time of 1.3 ms (760 Hz rate for both sliding mode controllers and both observers on a single cpu) limited the bandwidth of the observers by stability considerations of the numerical time integration of the observed plant simulation. Another limitation was due to the discretized measurement of link angles. With discretization, a smooth position ramp (i.e., a constant angular velocity) is perceived as a "staircase" position waveform. As a consequence, when the rate of measured position increments is below the observer's natural frequency, the observer behaves poorly. Between steps of the position measurement staircase function, the observer responds to an apparent sudden halt of the link position (no information is available between position increments). At the next position increment, the observer reacts to an apparent sudden change in velocity. The staircase position function can result in ringing of the velocity estimate, especially at encoder increment rates near the natural frequency of the observer error dynamics. When position increments occur rapidly with respect to observer error dynamics, then a smooth velocity estimate is obtained.

In the present application, accurate velocity estimates were required to evaluate the function $\tilde{s}$, which includes a term in ω^2. The ω^2 term is especially sensitive to estimate errors when ω is large. Thus, with regard to sliding-mode optimal-control implementation, ringing of the velocity estimates is more problematic at high frequencies than at low frequencies.

Discretized position measurements and resulting velocity estimate ringing are suspected as the source of torque chatter in figure III.5, the measured torque history for a sliding-mode optimal-control transient move. The observer parameters of $\omega_n = 100$ Hz and $\varsigma = 0.7$ were chosen as a compromise between high-speed error dynamics, desired for the linear-regime control, and the sensitivity of the sliding-mode controller to high-frequency velocity estimate ringing.

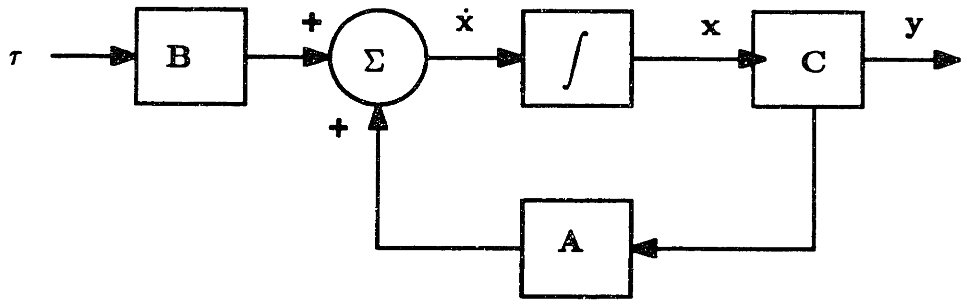

Figure 1: System Block Diagram

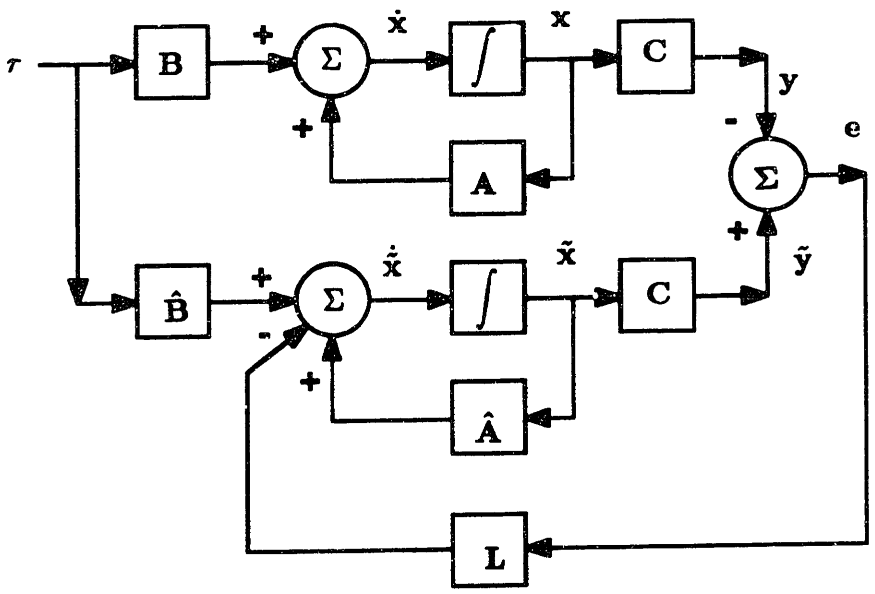

Figure 2: System plus Observer Block Diagram

BIBLIOGRAPHY

[1] Andrews,J.R. and Hogan,N., "Impedance Control as a Framework for Implementing Obstacle Avoidance in a Manipulator",in Control of Manufacturing Processes and Robotic Systems, Eds David Hardt/Wayne Book, A.S.M.E. Boston 1983, pp. 243-251.

[2] Asada,H. and Kanade,T., "Design of Direct-Drive Mechanical Arms", ASME J. of Vibration, Acoustics, Stress and Reliability in Design, Vol.105-3, 1983, pp. 312-316.

[3] Asada,H. and Youcef-Toumi,K., "Analysis and Design of a Direct-Drive Arm With a Five-Bar-Link Parallel Drive Mechanism", A.S.M.E. J. Dynamic Systems, Measurement and Control, Vol. 106, No. 3, 1984, pp. 225-230.

[4] Asada,H., Youcef-Toumi,K., and Ramirez,R., "Design of M.I.T. Direct-Drive Arm", Int. Symp. on Design and Synthesis, Japan Soc. of Precision Engineering. Tokyo, Japan, July, 1984.

[5] Asada,H., and Ro,I.H., "A Linkage Design for Direct Drive Robot Arms", A.S.M.E. J. of Mechanisms, Transmissions, and Automation in Design, Vol. 107, Dec. '85, pp. 536-540.

[6] Asada,H., and Slotine, J.J., ROBOT ANALYSIS AND CONTROL, J.Wiley, N.Y., 1986, Chp 5,6.

[7] Barbera, A.J., Fitzgerald.M.L., Albus,J.S., and Haynes,L.S., "RCS: The NBS Real-Time Control System", Proc. of Robots VIII Conference, Detroit, June,1984, pp. 19.1-19.33.

[8] Bobrow,J.E., Dubowsky,S. and Gibson,J.S., "On the Optimal Control of Robotic Manipulators with Actuator Constraints,", Proceedings of the 1983 Automatic Control Conference, pp.782-787 (June 1983).

[9] Book,W.J., and Majette,M., "Controller Design for Flexible, Distributed Parameter Mechanical Arms Via Combined State Space and Frequency Domain Techniques", A.S.M.E. J. Dynamics Systems, Measurement and Control, Vol. 105, Dec. 1983, pp. 245-254.

[10] Brooks,R.A., "A Robust Layered Control System For A Mobile Robot", IEEE J. Robotics and Automation, Vol. RA-2, No.1, March, 1986, pp.14-23.

[11] Brooks,R.A., "A Hardware Retargetable Distributed Layered Architecture for Mobile Robot Control", IEEE Int. Conf. on Robotics and Automation, April, 1987, pp. 106-110.

[12] Cannon,R.H., and Schmitz,E., "Initial Experiments on the End-Point Control of a Flexible One-Link Robot", Int. J. Robotics Research, Vol. 3, No. 3, 1984, pp. 62-75.

[13] Curran,R. and Mayer,G., "The Architecture of Adept One Direct-Drive Robot, Proc. American Control Conference, Boston, MA, June 19-21, 1985, pp. 716-721.

[14] Dubowsky,S., Norris,M.A, and Shiller,Z. "Time Optimal Trajectory Planning for Robotic Manipulators with Obstacle Avoidance: A CAD Approach", IEEE Int. Conf. on Robotics and Automation, April, 1986., pp. 1906-1912.

[15] Dupont,P.E., and Derby,S., "Planning Collision Free Paths for Redundant Robots Using a Selective Search of Configuration Space",A.S.M.E. Design Technical Conference, Oct 5-8, 1986, Columbus, Ohio, 86-DET-145.

[16] Eppinger,S.D., and Seering,W.P., "Understanding Bandwidth Limitations in Robot Force Control", IEEE Int. Conf. on Robotics and Automation, April, 1987, pp. 904-909.

[17] Erdmann,M., and Lozano-Perez,T., "On Multiple Moving Objects", M.I.T. Artificial Intelligence Laboratory Memo No. 883, M.I.T., Cambridge, Mass, May, 1986.

[18] Jacobson,S.C., Wood,J.E., Knuti,D.F., Biggers,K.B., and Iverson, E.K. "The Version I Utah/MIT Dextrous Hand", Proc. 5th CISM-IFToMM, June, 1984, pp. 1-8.

[19] Kent,E.W., Shneier,M.O., and Lumia,R., "PIPE (Pipelined Image Processing Engine)", J. Parallel and Distributed Computing, 2, 50-78, 1985.

[20] Khatib,O. "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots", J. Robotics Research, Vol 5, No 1, (Spring 1986),pp. 90-98.

[21] Khosla,P.K., "Real-Time Control and Identification of Direct-Drive Manipulators", Ph.D. thesis, Carnegie-Mellon University, Dept. of Electrical and Computer Engineering, Pittsburgh, Penn., August, 1986.

[22] Kirk,D.E., OPTIMAL CONTROL THEORY, Prentice-Hall, Englewood Cliffs, NJ, 1970, pp. 240-259.

[23] Koditschek,D.E., "Exact Robot Navigation by Means of Potential Functions: Some Topological Considerations", IEEE Int. Conf. on Robotics and Automation, April, 1987, pp. 1-6.

[24] B. Krogh, "A Generalized Potential Field Approach to Obstacle Avoidance Control",SME Conf. Proc. Robotics Research: The Next Five Years and Beyond, Bethlehem, Pennsylvania, August, 1984.

[25] Lozano-Perez,T., and Wesley, M.A., "An Algorithm for Planning Collision Free Paths Among Polyhedral Obstacles", Communications of the A.C.M., Vol.22, No.10, Oct. 1979, pp. 560-570.

[26] Lozano-Perez,T. "Spatial Planning: A Configuration Space Approach,", IEEE Trans. Computers, Vol.C-32, No.2, February, 1983, pp. 108-120.

[27] Luenberger,D.G., "An Introduction to Observers", IEEE Transactions on Automatic Control, AC-16, No. 6, Dec. 1971, pp. 596-602.

[28] Lumelsky, V., Stepanov, A., "Dynamic Path Planning for a Mobile Automaton with Limited Information on the Environment", IEEE Transaction on Automatic Control, Vol.AC-31, No.11,November 1986.

[29] Lumelsky, V., "Continuous Motion Planning in Unknown Environment for a 3-D Cartesian Robot Arm", Proc. IEEE Int. Conf. on Robotics and Automation, April 1986,pp. 1050-1055.

[30] Lumelsky,V., "On the Path Length Performance of Maze-Searching Algorithms", Yale University, Center for Systems Science, New Haven, CT, Report No. 8710, July 1987.

[31] Makino,H. et al., "Research and Development of the SCARA Robot", Proc. of the 4th Int. Conf. on Production Engineering, Tokyo, 1980, pp. 885-890.

[32] Meckl,P. and Seering,W., "Active Damping in a Three-Axis Robotic Manipulator", A.S.M.E. J. of Vibration, Acoustics, Stress, and Reliability in Design, Vol. 107, Jan. 1985, pp. 38-46.

[33] Myers, J.K., "Multiarm Collision Avoidance Using the Potential-Field Approach",Proc. SPIE Int. Soc. Opt. Eng. (USA), vol.580, 78-87, 1985.

[34] Newman,W.S. and Hogan,N., "Time Optimal Control of Balanced Manipulators", A.S.M.E. Winter Annual Meeting, Anaheim, CA, November, 1986.

[35] Newman,W.S., and Hogan,H., "High Speed Robot Control and Obstacle Avoidance Using Dynamic Potential Functions", IEEE Int. Conf. on Robotics and Automation, April, 1987, pp. 14-24.

[36] Pasch,K.A. and Seering,W.P., "On the Drive Systems for High-Performance Machines", A.S.M.E. J. Mechanisms, Transmissions and Automation in Design, Vol. 106, No. 1, March '84, pp. 102-108.

[37] Paul,R.P., and Zhang,H., "Computationally Efficient Kinematics for Manipulators with Spherical Wrists Based on Homogeneous Transformation Representation", Int. J. Robotics Research, Vol. 5, No. 2, 1986, pp. 32-44.

[38] Pieper,D.L., "The Kinematics of Manipulators Under Computer Control", Ph.D. thesis, Dept. of Computer Science, Stanford University, Stanford, CA., 1968.

[39] Red,W.E., and Truong-Cao, H.V., "Configuration Maps for Robot Path Planning in Two Dimensions", A.S.M.E. J. Dynamics Systems, Measurement and Control, Vol. 107, Dec. 1985, pp. 292-298.

[40] Sahar,G. and Hollerbach,J.M., "Planning of Minimum-Time Trajectories for Robot Arms", A.I. Memo No.804, MIT Artificial Intelligence Laboratory, November 1984.

[41] Salisbury,J.K., "Design and Control of an Articulated Hand", Proc. 1st Int. Symp. on Design and Synthesis, Tokyo, July 1984.

[42] Schmitz,D., Khosla,P.K. and Kanade,T., "Development of CMU Direct-Drive Arm II", Proc. 15th Int. Symp. on Industrial Robotics, Tokyo, Japan, Sept. 11-13, 1985.

[43] Schwartz,J.T. and Sharir,M., "On the Piano Movers Problem 1: The Case of a Two-Dimensional Rigid Polygonal Body Moving Amidst Polygonal Barriers", Dept of Computer Science, Courant Institute of Mathematical Sciences, New York University, 39 (1981).

[44] Schwartz,J.T. and Sharir,M., "On the Piano Movers Problem 1: General Properties for Computing Topological Properties of Real Algebraic Manifolds", Dept of Computer Science, Courant Institute of Mathematical Sciences, New York University, 41 (1982).

[45] Shin,K.G. and McKay,N.D., "Minimum-Time Control of a Robotic Manipulator with Geometric Path Constraints", IEEE Transactions on Automatic Control, AC-30(6), (June 1985).

[46] Slotine,J.J., and Sastry,S.S.,"Tracking Control of Non-linear Systems Using Sliding Surfaces, with Application to Robot Manipulators", Int. J. Control, 1983, Vol. 38, No. 2., pp. 465-492.

[47] Slotine,J.J., "The Robust Control of Robot Manipulators", Int. J. of Robotics Research, Vol 4., No 2, 1985, pp. 49-64.

[48] Takegaki,M. and Arimoto,S. "A New Feedback Method for Dynamic Control of Manipulators", A.S.M.E. J. Dynamic Systems Measurement and Control, Vol. 102, (June 1981), pp. 119-125.

[49] Udupa,S., "Collision Detection and Avoidance in Computer Controlled Manipulators", Ph.D. dissertation, Dep. Elec. Eng. California, Inst. of Technol., 1977.

[50] W. M. Wonham, LINEAR MULTIVARIABLE CONTROL: A GEOMETRIC APPROACH, Springer-Verlag, NY, 1979, pp. 116-122.

[51] Wu, C.H., and Paul, R.P., "Manipulator Compliance Based on Joint Torque Control", Proceedings 19th IEEE Conf. on Decision and Control, Vol. 1, Dec. 9-12, 1980, pp. 88-94.

[52] K. Youcef-Toumi, "Analysis, Design and Control of Direct-Drive Manipulators", Ph.D. thesis, MIT Dept. of Mechanical Engineering, May 1985.

[53] Young,K.K.D., "Controller Design for a Manipulator Using Theory of Variable Structure Systems", IEEE Trans. Syst. Man Cybernetics, SMC-8, 2, 1978, pp. 101-109.