Aggressive Trajectory Generator for a Robot Ping-Pong Player

Russell L. Andersson

ABSTRACT: Robot trajectory planning algorithms often ignore the dynamics of the robot and actuator to concentrate on other aspects, such as Cartesian linearity or collision avoidance. This paper describes a trajectory generation and tracking system designed to maximize the manipulator's usable performance on a sensor-driven task. The system drives a PUMA 260, which is part of a functioning robot ping-pong player. We will describe the system's implementation and performance in practice.

Introduction

In this paper, we will describe a trajectory generation and tracking system designed to operate in a very dynamic environment. We will try to achieve predictable, and reliable, high-performance motions at the envelope of robot capabilities, unlike most systems, which plan conservative motions to avoid having to consider actual manipulator capabilities.

We assume the task is sensor-driven, such that each motion is unique, and that motions will have to be changed while they are in progress. Approaches based on learning or global knowledge of the trajectory are thus ruled out.

We will be concerned with free-space arm motion, which is not subject to task-induced constraints on the interior of the trajectory (not painting or force-controlled applications). However, the trajectory may have arbitrary initial and final conditions; especially, the robot's "final" velocity may be nonzero. We also include the possibility that the motion must terminate by a specific time, which facilitates interaction with a changing environment.

Our primary application was to control a robot, shown in Fig. 1, that plays ping-pong [1] according to international robot rules [2]. To play a reasonable game against human opponents, the system uses a 60-Hz, three-dimensional stereo-vision system as input, a specialized real-time expert controller, and the robot control system described herein. Similar possible applications include: manipulating objects on moving conveyor belts, interacting with fixed-cycle-time devices (machine tending), scheduling multirobot systems, or other applications where performance is at a premium.

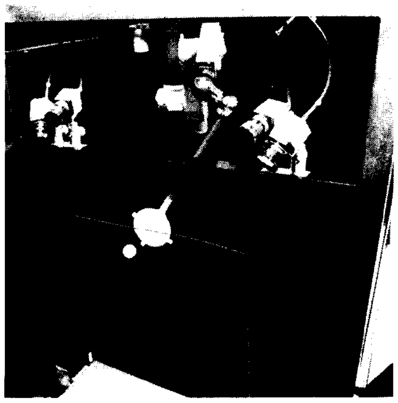
Fig. 1. Robot ping-pong player.

We consider the robot task planner to be divided into two stages: an upper task-level controller and the lower-level robot-specific controller. In this paper, we will be concerned with the lower-level control and its interface to the higher-level control, but not the higher task-level control itself.

We assume that the task has several degrees of freedom available for modification in response to robot-related conditions. In the simplest case, we might have the motion time as a degree of freedom, such that a motion's duration can be increased or decreased in response to the stress it places on the manipulator's actuators. More complex tasks may have internal degrees of freedom, or even robots with redundant joints; the same end effect may be achieved by allocating the motion among the different degrees of freedom. For example, the robot ping-pong player can vary the position along the ball's trajectory at which it will be hit, the direction in which the ball will be hit, and the orientation of the paddle handle. (Ping-pong requires only five degrees of freedom, though the robot has six.)

The task-level control determines the manipulator's trajectory by selecting values for the degrees of freedom under its control. The task-level control has little choice but to "guess," using all the guile it can command, at a suitable trajectory, in the hope that the manipulator will be able to execute it, and that the trajectory does not "waste" the machine's capabilities by understressing it.

Once the task-level control has selected values for the degrees of freedom and generated the corresponding trajectories (using low-level support routines), the lower-level control must evaluate the trajectories' feasibility, and either execute the trajectories or indicate that they are infeasible. In either case, the lower-level control must provide quantitative information to the task-level control, which will enable the latter to improve the trajectories it is requesting—to increase or decrease their stress level as appropriate. The feedback process is essential to making aggressive use of a manipulator. A conventional system might simply blow a fuse or surreptitiously overshoot the target. As described, we want high performance that is reliable.

In this paper, first we will describe some of the requirements in more detail, compare alternative high-performance trajectory types, describe the properties of the quintic polynomials we use, discuss how to determine if a trajectory is feasible, and then outline our implementation and its performance.

Requirements

Let us examine the requirements on the lower-level control system and the trajectories it generates in more detail. We will do our planning in joint space (rather than Cartesian, for example), because the limits on the robot's performance are expressed primarily in this space. Cartesian motions are required mainly when the robot is interacting with the environment; such interaction tends to place external restraints on the robot's performance, i.e., painting must be done at a controlled rate. We are interested primarily in the unrestrained motions, defined by their initial and final states, where maximal performance is required. It is these motions whose performance is most critical,

An early version of this paper was presented at the 1988 IEEE International Conference on Robotics and Automation, Philadelphia, Pennsylvania, April 24-29, 1988. Russell L. Andersson is a Member of the Technical Staff at AT&T Bell Laboratories, Holmdel, NJ 07733.

because they take most of the time. When necessary, the system can generate Cartesian motions with appropriate combinations of joint-space motions—this is how Cartesian motions are ultimately implemented in any case.

Continuity

The trajectory should be continuous in position, velocity, and acceleration. A continuous acceleration not only reduces the generation of higher-order vibrations in the arm, but prevents the task-level control from believing that the acceleration can be changed instantaneously.

Malleability

As new sensor data arrive, we will be constantly modifying the terminal characteristics of the trajectory. The old trajectory must be continuous with the new trajectory and must be well-behaved even for substantially different targets. We must be able to predict the parameters of the transition. Simple transition-window schemes, which linearly interpolate between one trajectory and another, are not satisfactory. In some cases, the time between alterations may be even less than any reasonable transition time. The robot ping-pong task generates new trajectories every 50 msec.

Notice that we do *not* assume that the system simply servos at a sensor-supplied position, which does not require trajectory planning. Sensor-induced updates may arise very early in the motion when the manipulator is still close to the starting point; the trajectory planner must smoothly alter its destination. Sensors may even indicate that an entirely different sequence of motions may be necessary, so we must be able to entirely retract a planned motion if needed.

Trajectory Description

To maintain continuity of position, velocity, and acceleration across sensor-induced updates, the trajectory representation must include the following parameters: initial position p_i, initial velocity v_i, and initial acceleration a_i. To be useful in describing the desired final state, the trajectory representation also requires these parameters: final position p_f, final velocity v_f, final acceleration a_f, and time of flight t_f. In our case, the final acceleration, a_f, always will be zero, because we are trying to attain a designated final velocity and position. Thus, by maintaining the final acceleration at zero, we minimize the sensitivity of the final velocity to slight errors in the final time. The trajectory specification requires seven internal degrees of freedom, including the motion duration, to be able to meet these boundary conditions.

Most robots assume that they should be stopped at the final position. By allowing a nonzero final velocity, we enable the direct specification of a motion's intermediate way points (''continuous-path'' mode).

Trajectory Feasibility

As mentioned in the introduction, the lower-level control must be able to establish a proposed trajectory's feasibility, which requires consideration not only of the robot dynamics, but the actuator characteristics as well. This contrasts with earlier work [3] that assumes arbitrary velocity limits, but we restrict the results to a specific actuator model.

The torque generated by a motor is proportional to the current through it [4]. The motor winding power dissipation and the demagnetization flux of the motor limit the peak current and, thus, torque, to some upper limit, T_{clip}.

During motion, the motor behaves as a generator, creating a voltage or electromotive-force (back-EMF) that opposes further acceleration in that direction. However, once the voltage range of the amplifier has been reached, the motor can generate only a reduced torque. The maximum voltage is limited by the motor winding insulation and the amplifier's power transistors. As the velocity increases, the achievable torque drops until, at maximum velocity, no torque can be generated.

Although the PUMA 260's specification lists 1.0 m/sec as the maximum tool velocity, the back-EMF-limited joint speeds range from 350 deg/sec for J2 (joint 2) to 2000 deg/sec for J6. Spinning J1 at its maximum rate with the arm outstretched would yield a wrist velocity of 3.5 m/sec. We would like the lower-level control to be able to take advantage of this performance. To do so, we need good models and time-efficient algorithms to determine feasibility.

Acceleration-Minimizing Position and Velocity

If the maximum actuator performance limit is being approached, the task-level control needs to know how to modify a proposed trajectory to reduce its acceleration requirements. Given a specified motion, we must know: what final position would minimize the maximum acceleration (leaving the final velocity unchanged), and what final velocity would minimize the maximum acceleration (leaving the final position unchanged)? When the velocity or acceleration boundary conditions are nonzero, the intermediate trajectory can be surprising, making the acceleration-minimizing parameters nontrivial.

Comparison of Trajectory Types

The shape of the canonical trajectory is a crucial decision; it must meet the constraints outlined in the previous section and be computationally tractable. A fundamental measure of a trajectory type is how far it can drive the robot in a certain period of time. Figure 2 compares four trajectory types by displaying the maximum displacement that may be obtained by starting from rest, moving, and returning to rest. Joint 6, on which the plot is based, has a range of 530 deg. The trajectory types, in order of the distance traveled in 0.5 sec from farthest to shortest, are: full power, trapezoidal velocity profile, quintic polynomials with back-EMF avoidance, and quintic polynomials with worst-case back-EMF assumptions.

Full Power

Let us consider applying full power to the motor, allowing it to ramp up to the back-EMF limit, and then, at just the right instant, reversing the torque and decelerating at the maximum rate to a stop just at the end of the allowed time interval. Note that we can always decelerate at the full rate because the back-EMF is helping rather than hindering the amplifier. The resulting plot reflects the maximal performance of the robot, motor, and amplifier, giving a benchmark for further comparisons.

However, this scheme does not have enough degrees of freedom, is maximally discontinuous, and requires iterative approaches to identify key parameters. It has no provision for servoing during the course of execution, and, in addition, since all of

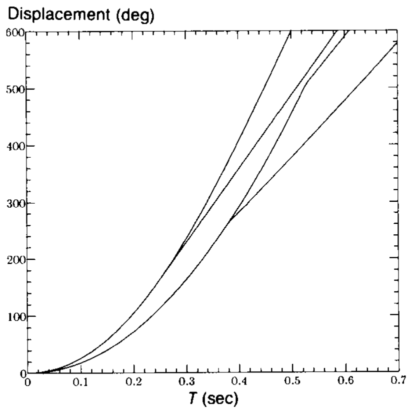

Fig. 2. Comparison of trajectory types by distance traveled as a function of motion time.

the actuator torque already has been used in one direction, there is no available correction if the actuator fails to accelerate as fast as expected.

Trapezoidal Velocity Profile

A somewhat restrained version of the full-power approach is to accelerate at the maximum rate, run at a constant, servoed, velocity as soon as the back-EMF limit is encountered, and then decelerate at maximum torque until the end of the motion, resulting in a trapezoidal velocity profile. This is the second highest curve in Fig. 2.

To be applied, the acceleration must be smoothed by adding linear transition regions at acceleration discontinuities, complicating implementation, and reducing performance from that shown in the graph. The allowable acceleration must be derated at lower speeds to allow a margin for servoing.

The full generalization of this approach for arbitrary initial and final conditions would require a large number of cases and frequent switching between trajectory segments. The acceleration would not always be equal to the maximum in this case.

Quintic Polynomials

Polynomials are a natural choice for a motion representation because their properties are well known, and closed-form solutions may be derived for many of their properties. The derivatives are very easy to compute and may be represented in the same form.

To match the six degrees of freedom required for trajectory planning, we will use fifth-order, or quintic, polynomials. Others have proposed the use of quintic polynomials and cubic splines [5], [6]. Only a single quintic polynomial is needed to represent the entire trajectory, so the trajectory is inherently very smooth and easy to evaluate. Quintic polynomials do not require the overhead of multiple subsegments, as would cubic splines. Cubic splines also would require an automated way of picking intermediate knot points and conditions.

A quintic trajectory is shown in Fig. 3. As may be seen from the figure, the acceleration does not remain at its peak value for very long. There are several important consequences to this.

First, we cannot go as far in a given amount of time; the bottom curve in Fig. 2 is a quintic. However, since we do not count as much on the trajectory's peak acceleration, we need not derate the motor's peak acceleration to leave room for servoing but can take full advantage of the computed peak motor acceleration.

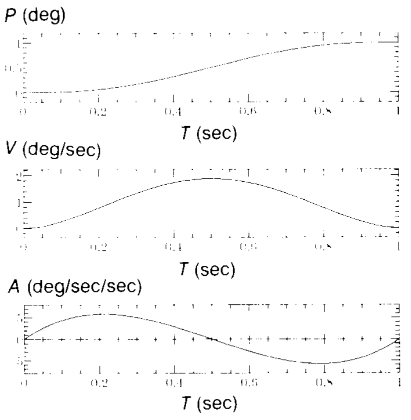

Fig. 3. Example of a quintic trajectory.

Another examination of Fig. 3 shows that, by the time the velocity curve has reached substantial levels, the acceleration has dropped. We do not need to do back-EMF planning based on the maximum velocity but can "shoehorn" a larger acceleration under the back-EMF limit. Consequently, the performance of a well-planned quintic can approach that of the trapezoidal and full-power profiles, as shown by the third highest curve in Fig. 2.

The quintic attains 70 to 80 percent of the displacement of the trapezoidal curve for durations between 0.2 and 0.4 sec. By 0.525 sec, the two curves are nearly equal. If T_{clip} is more than 47 percent of $T_{\max}$ (see the following subsection, "Motor Model," for exact definitions), the performance of the quintic will exceed that of the trapezoidal over some portion of the trajectory. Because the trapezoidal trajectory generator must smooth its acceleration curve to reduce jerk, the actual trapezoidal curve is even lower than that shown, reducing any advantage of the trapezoidal approach near zero.

In light of the quintic approach's simplicity and good performance, it appears to be an excellent choice. The task-level control specifies the trajectory for each joint with a start time, a quintic polynomial, and a duration. The start and duration times are referenced to the system's global clock so that the manipulator can interact with dynamic environments. Lower-level-control support routines simplify the generation of the trajectory and times. A temporally sorted queue arrangement in the lower-level control allows several quintics to be concatenated to represent different stages of the task. When new sensor data cause quintics to be added to the queue, we flush any part of the planned trajectory that occurs after the new quintic's start.

Properties of Quintic Polynomials

In this section, we will describe properties and uses of quintic polynomials that pertain to robot trajectory planning and derive some useful results. Given a basic quintic polynomial:

$$p(t) = a_5t^5 + a_4t^4 + a_3t^3 + a_2t^2 + a_1t + a_0 \qquad (1)$$

we can match arbitrary initial and final conditions with these coefficients:

$$\begin{aligned} a_5 &= \{t_f[(a_f - a_i)t_f - 6(v_i + v_f)] + 12(p_f - p_i)\}/(2t_f^5) \\ a_4 &= \{t_f[16v_i + 14v_f + (3a_i - 2a_f)t_f] + 30(p_i - p_f)\}/(2t_f^4) \\ a_3 &= \{t_f[(a_f - 3a_i)t_f - 8v_f - 12v_i] + 20(p_f - p_i)\}/(2t_f^3) \\ a_2 &= a_i/2 \\ a_1 &= v_i \\ a_0 &= p_i \end{aligned} \qquad (2)$$

A fully matched trajectory is shown in Fig. 4.

It is helpful to be able to determine the minimum and maximum values of the trajectory's derivatives. The acceleration limits predict trajectory feasibility—a small velocity limit can eliminate the need to check for back-EMF problems, and the position limits can aid collision prevention and keep the joint from going past its limit stops. The maximum and minimum accelerations may be found easily by solving a quadratic, but the velocity and position limits require cubic

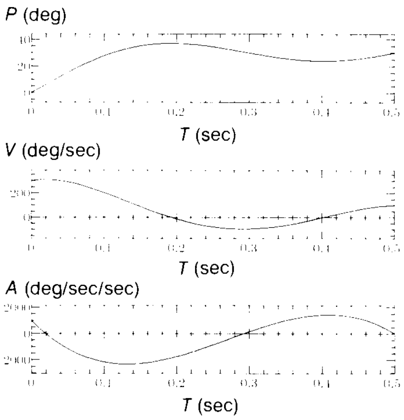

Fig. 4. Quintic with matched position, velocity, and acceleration.

and quartic roots, respectively (which can be computed in closed form).

For a motion with zero initial and final velocities and accelerations,

$$v_{\max} = 15(p_f - p_i)/(8t_f) \tag{3}$$

$$a_{\max} = \pm 5.77(p_f - p_i)/(t_f^2) \tag{4}$$

From these calculations, we can see that the acceleration is (predictably) quite sensitive to the time of flight. The peak velocity is slightly under a factor of 2 greater than the average velocity.

Acceleration-Minimizing Final Position

Suppose that a joint is near its maximal acceleration, and we wish to find the final position that minimizes the acceleration. Knowing this position, the task-level control can attempt to alter the problem so that this final position is attained, thus minimizing the stress on the joint.

A straightforward attempt to compute the maximum acceleration as a function of an arbitrary p_f, and then to minimize this maximum over p_f turns out to be infeasible. Instead, we choose an arbitrary time, t_j, to be an acceleration maximum, and then find the t_j that minimizes the acceleration at that time. The equations then turn out to be solvable. We will not show all of the intermediate results because of space limitations but will outline the procedure involved.

To force the acceleration maximum to a particular time, we use a constraint that sets the jerk ($\dot{a}$) to be zero:

$$j(t_j) = 60a_5t_j^2 + 24a_4t_j + 6a_3 = 0 \tag{5}$$

With this constraint and the five initial and final conditions (all but p_f), we can solve for the six polynomial coefficients. Substituting the coefficients into the acceleration polynomial and evaluating it at $t = t_j$, we get the quantity to be minimized; namely, the greatest acceleration $a_{\max}$. We then evaluate

$$\partial a_{\max}/\partial t_j = 0 = \mathbf{P}(t_j)/\mathbf{Q}(t_j) \tag{6}$$

such that **P** and **Q** are polynomials. We can ignore **Q** and factor **P**. The factors specify the times of interest t_j and include 0, t_f, $t_f/2$, and a lengthy unfactorable quadratic in t_j. We choose to ignore the unfactorable polynomial altogether, although there is some chance that it might produce roots in the region of interest. If the initial acceleration $a_i = 0$, then the quadratic is considerably simplified—to one that has no real roots. Since the acceleration is prespecified at the end points of the interval (0 and t_f), those roots are not of interest. The root at $t_f/2$ is the root of interest.

The resulting acceleration-minimizing position is

$$p_{a,\min} = \{10t_f(v_i + v_f) + (a_i - a_f)t_f^2\}/20 + p_i \tag{7}$$

The peak acceleration is

$$a_{p,\min} = \{6(v_f - v_i) - (a_i + a_f)t_f\}/(4t_f) \tag{8}$$

Acceleration-Minimizing Final Velocity

Similarly, we can find the final velocity, which minimizes the acceleration required to get to a final position at a specified time. This time we find the important root at $t_j = 2t_f/5$.

The resulting acceleration-minimizing velocity is

$$v_{a,\min} = \{44(p_f - p_i) - 20v_it_f + (3a_f - a_i)t_f^2\}/(24t_f) \tag{9}$$

The peak acceleration is

$$a_{v,\min} = \{72(p_f - p_i) - 72v_it_f - (9a_i + 2a_f)t_f^2\}/(25t_f^2) \tag{10}$$

Trajectory Feasibility Determination

During the planning process, we must establish if a given proposed trajectory is feasible. By feasible, we mean: can the robot actuators cause the robot to follow this trajectory while still maintaining the accuracy required for the task? Knowledge of the manipulator's capabilities is essential to being able to drive the robot at the limits of those capabilities. We must have a good robot dynamics model.

If the task-level control plans an infeasible trajectory, the lower-level control will point this out by describing the problem's severity and providing facilities to aid the trajectory's modification. The task-level control must change the planned motion time or final characteristics, taking advantage of redundant degrees of freedom in the task. If the motion must end at a specific time, the task-level control must take into account the reduced available motion time; it must act quickly to avoid a spiral into paralysis.

Robot Total Dynamics

The torque at a manipulator joint may be written as shown, where D_{ij} are the coupling inertias and D_{ii} includes the effect of the actuator inertia (Paul [6], with additions):

$$T_i = \sum_{j=1}^{6} D_{ij}a_j + \sum_{j=1}^{6}\sum_{k=1}^{6} D_{ijk}v_jv_k + D_i - F_iv_i - S_i \operatorname{sgn}(v_i) \tag{11}$$

The acceleration of joint j is a_j; D_{ijk} are centripetal and Coriolis terms; v_j is the velocity of joint j; D_i is the gravitational load on joint i; F_i is the viscous damping coefficient for joint i; and S_i is the intercept of the friction at rest (not the same as static friction, see Zhang [7]). The static friction is larger, but we describe only the moving robot.

Motor Model

The motor torque/speed characteristics may be described by $T_{\max}$, T_{clip}, and k, such that the torque on the joint [T_i from Eq. (11)], as a function of the joint velocity v, is

$$\max(-T_{\text{clip}}, -T_{\max} - kv) \le T_i \le \min(T_{\text{clip}}, T_{\max} - kv) \tag{12}$$

The parameters are computed from the motor and amplifier characteristics by the following, where g is the gear ratio, K_m the motor torque constant, $I_{\max}$ the maximum allowable motor current, $V_{\max}$ the maximum amplifier voltage, R the motor winding resistance, and K_e the back-EMF constant (V-sec/rad):

$$\begin{aligned} T_{\text{clip}} &= gK_mI_{\max} \\ T_{\max} &= gK_mV_{\max}/R \\ k &= g^2K_mK_e/R \end{aligned} \tag{13}$$

Back-EMF Avoidance

We would like to be able to compute the worst-case margin (i.e., the smallest) between the acceleration we are planning to demand for each joint, a_i, and the acceleration that can be generated by the motors. If the margin is less than zero, the motion is unacceptable and must be replanned.

We can rewrite Eq. (12) to the following equation by substituting in Eq. (11) (considering only the positive limits here), and then casting out terms:

$$a_{\text{margin}} = \{[\min(T_{\text{clip}}, T_{\max} - kv) - \sum_{j \ne i} D_{ij}a_j - D_i + F_iv]/(D_{ii})\} - a \tag{14}$$

Since the coefficients D_i and D_{ij} change dramatically with position, we approximate by taking the worst-case value of these parameters at the beginning and end of the trajectory. We clearly throw away system performance by this simplification and, furthermore, run the risk of not catching actual limits. Once we have done this, the minimum a_{margin} may be found by computing its coefficients as a fourth-order polynomial, then solving the cubic to find candidate minima. A quadratic checks the minimum and maximum acceleration against T_{clip}.

Note that we have dropped the terms $D_{ijk} v_i v_k$ from Eq. (14) because they cause eighth-order equations, and we are not computing D_{ijk} anyway. The friction intercept term (S_i) is nonlinear and not that significant, so we have dropped it also. In practice, we currently ignore the coupling inertias and friction terms in Eq. (14) to the detriment of accuracy and the benefit of computational speed, although both terms are conceptually straightforward.

The ratio of the acceleration margin to the maximum motor acceleration, $D_{ii} a_{\text{margin}}/T_{\text{clip}}$, is of great importance to the task-level control because it indicates the degree of stress the joint is subjected to.

Implementation and Performance

Our controller architecture (Fig. 5) uses multiple slaves to off-load the main processor, which runs the MEGLOS operating system [8]. The S/Net interface connects to other processors that perform sensing and analysis. The need for a global wall clock has been discussed by Andersson [9]. It allows us to maintain accurate timing across a network of processors.

A slave processor computes the kinematics and dynamics—Cartesian position, inertias, coupling torques, and gravities. Polynomial interpolation and position and derivative control occur in the joint processors. The processors are connected via shared memory, enabling the main processor to monitor the slaves transparently.

We feed-forward as many torques as possible, so that only torques that are not fed-forward affect the accuracy [10]. From Eq. (11), we compute gravity loads and D_{ii} inertias at the 26-msec major cycle rate. We also compute the torque due to the coupling pseudoinertias D_{ij} for $i \neq j$ each 26 msec by using the acceleration at the midpoint of the 26-msec cycle. (At present, only the coupling inertias to and from the wrist joints are computed. There are no terms among joints 1, 2, and 3, even though there are terms from joints 1 to 6, for example.) The equations are based on those by Izaguirre and Paul [11]. At present, we do not feed-forward centripetal and Coriolis effects. Timing analysis indicates that the kinematics/dynamics processor will be able to handle the full dynamics; we have simply not implemented it yet.

The joint servos execute the position and derivative control routine every 830 μsec, computing the feedforward due to friction and acceleration (of that joint alone), then adding the feedforward for that 26-msec cycle. We add an integrating term to the position and derivative control when the desired velocity is zero to minimize the steady-state error, but disable the integrator during motion because the trajectories are so dynamic that the integrator is a hindrance.

By computing the feedforward torque, inertia, and gravity only at a much slower rate, we greatly reduce the computation required of the joint servo and the amount of data that needs to be transferred among the processors. The side effect is a small increase in the apparent disturbing torque.

Let us take a look at the robot's performance on the ping-pong task by using live motions extracted from a data-logging system. The paddle is mounted at the end of a 0.45-m stick to increase the robot's reach, and so that a reasonable rotation rate for joint 6 can generate the striking velocity. This geometry creates a large load inertia, approximately 2 oz-in.-sec^2, greatly exceeding the manufacturer's specification of 0.5 oz-in.-sec^2. Unlike many dynamics papers, which make the simplifying assumption that the center of gravity of the load is along the axis of the last link, our dynamics are valid for arbitrary loads.

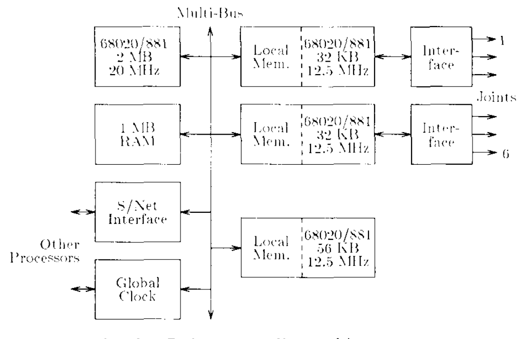

Fig. 5. Robot controller architecture.

Figures 6 and 7 show the joint velocities and velocity errors while executing one swing. Both the desired and actual velocities are overlaid in Fig. 6, although it is difficult to see. The motion lasted 0.3 sec. The peak paddle acceleration was 2.5 *g*; the peak velocity, 1800 mm/sec. At the contact time, the total position error was 3 mm, while the arm was moving at 1400 mm/sec. The effective timing accuracy was 2 msec. The position errors of all six joints deflected the paddle normal by 0.25 deg.

Joint 6 can be seen to build up substantial speed while positioning for the hit, then run briefly at constant velocity in the vicinity of the hit. The other joints come to a stop for the hit instant, then accelerate to return to the get-set position.

The error at the hit time corresponds to approximately 40 mm/sec. The errors could doubtless be improved by computing the full dynamics. A more subtle problem is that the 260's drive shafts (from the motors to the gears) contain flexible couplings, which act as torsional springs. This joint flexibility is difficult to compensate without simultaneous measurements of the joint and motor positions.

The system plays complete games against human opponents, keeping score and making commentary. The tight constraints posed by the robot's table frustrates novice and expert humans alike—accuracy defeats velocity. We have attained volleys of 21 hits by human and machine—a tribute to the performance of both. The system's fluid performance distracts even technical observers from the underlying issues.

Summary

The robot control package has emphasized the generation of accurate and predictable motions. The system is especially appropriate for sensor-driven tasks where motions may be changed while they are in progress, and when the robot must interact with rapidly changing environments.

The cooperative interaction between the task-level control and the lower-level control is essential to making aggressive use of the manipulator. The more autonomy the task-level control is allowed, the greater use it can make of the manipulator. Redundancy in the task, trajectories, and robot is beneficial, because it gives the task-level control more possibilities for improving the trajectories. All robots are redundant—we must exploit that redundancy.

By using quintic polynomials to generate trajectories, on short motions we can achieve 70 percent of the distance the joint would

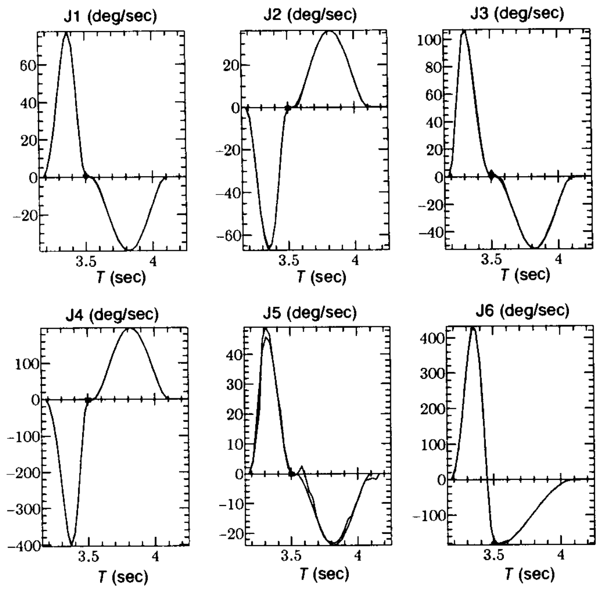

Fig. 6. Joint velocities during motion.

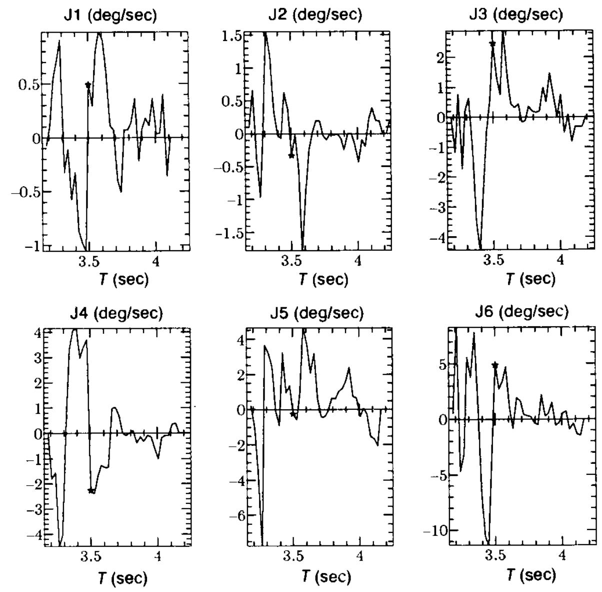

Fig. 7. Joint-space velocity errors.

travel if the motors were at maximum torque, while keeping the joint under planned, servoed control. The percentage is higher on long motions. At the same time, the regularity and consistency of the polynomial representation enable us to obtain a variety of useful results in closed form.

The system uses simplified dynamics to assess the feasibility of proposed trajectories. To consider the entire dynamics, we must basically simulate the trajectory. The lower-level control provides information that allows the task-level control to improve the trajectories. We designed the architecture so that the task-level control could monitor the torques during arm movement and tune the trajectory on the fly if the performance is less than expected. The comparatively slow rate of the task-level control's updates makes it unlikely that problems could be corrected at present, because a problem would probably not be detected until the acceleration was fairly extremal. A quick look-ahead simulation might ameliorate this difficulty.

The implementation's architecture offloads work to the slave processors, making almost all (on the order of 90 percent) of the main processor's time available for planning. As more processor cycles become available, we will continue to be able to increase the speed and accuracy of the arm's motion.

References

[1] R. L. Andersson, *A Robot Ping-Pong Player: Experiment in Real-Time Intelligent Control*, MIT Press, 1988.

[2] J. Billingsley, "Machineroe Joins New Title Fight," *Practical Robotics*, pp. 14-16, May/June 1984.

[3] W. D. Fisher and M. S. Mujtaba, "Minimum Ratio-Locked Profile Times for Robot Trajectories," *IEEE Intl. Conf. on Robotics and Automation*, vol. 2, pp. 1054-1060, Apr. 1988.

[4] "DC Motors, Speed Controls, Servo Systems," Electro-Craft Co., 1980.

[5] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, Addison-Wesley, 1986.

[6] R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control*, MIT Press, 1981.

[7] H. Zhang, "Design and Implementation of a Robot Force and Motion Server," Ph.D. Dissertation, Purdue Univ., West Lafayette, IN, May 1986.

[8] R. D. Gaglianello and H. P. Katseff, "MEGLOS: An Operating System for a Multiprocessor Environment," *Proc., Fifth Intl. Conf. on Distributed Computing Systems*, May 1985.

[9] R. L. Andersson, "Living in a Dynamic World," *Proc., ACM-IEEE Fall Joint Computer Conf.*, pp. 97-104, Nov. 1986.

[10] C. H. An, C. G. Atkeson, and J. M. Hollerbach, "Experimental Determination of the Effect of Feedforward Control on Trajectory Tracking Errors," *Proc., IEEE Intl. Conf. on Robotics and Automation*, vol. 1, pp. 55-60, Apr. 1986.

[11] A. Izaguirre and R. P. Paul, "Automatic Generation of the Dynamic Equations of the Robot Manipulators Using a LISP Program," *Proc., IEEE Intl. Conf. on Robotics and Automation*, vol. 1, pp. 220-226, Apr. 1986.

Russell L. Andersson was born in Philadelphia, Pennsylvania, and grew up in the Valley Forge area. He received the B.S. and M.S. degrees in computer science from the University of Pennsylvania in 1980 and 1981. He joined AT&T Bell Laboratories in 1981 and is currently a Member of the Technical Staff in the Ro-