TRAJECTORY CONTROL OF A COMPUTER ARM*

by

Richard Paul

Stanford Artificial Intelligence Project
Stanford University
Stanford, California USA

This paper describes the programming of a computer controlled arm. The programming is divided logically into planning and execution Communication between planning and execution is by a data file which specifies the arm trajectory with reapect to time, and actions that the arm should perform. The servo program which moves the arm along the trajectory is based on Legrangian mechanics and takes into account coupling between links, and the variation of inertial loading with change of arm configuration.

Key words: arm, trajectory, servo

INTRODUCTION

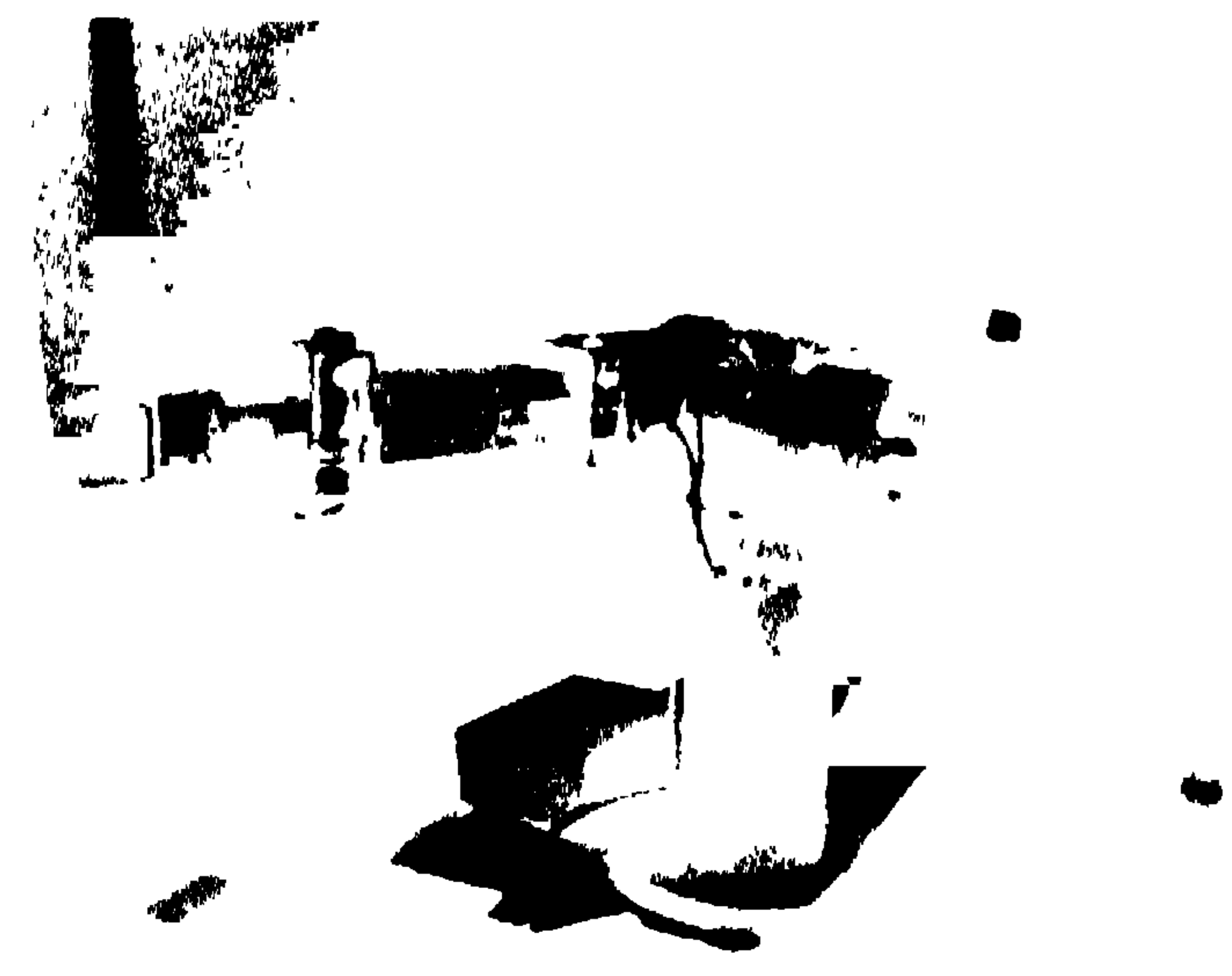
We are Interested in driving a computer controlled arm such as the one shown in Fig. 1. This arm [1] has six degrees of freedom with a vise grip hand and a useiuL working area about equivalent to that of a human arm.

The arm is powered by printed circuit eietric motors with harmonic drive gear reductions. The first two joints are rotary; they are followed by a prismatic joint and then by three rotary joints whose axes intersect.

Joint position is determined by six structurally integrated potentiometers read into the computer by a 12 bit analog to digital converter. Output is by means of a 9 bit digital to voltage pulse width converter producing a 15 millisecond pulse at maximum output. The servo program is executed by a PDP-6 computer which, as a part of the time sharing system, is availabJe for real time applications. The servo program is a sampled data system with a sample rate of 60 samples per second.

Information about the location of objects is derived from a vision programming system utilizing a TV camera as input [2] and is available as a global data structure.

*This research was supported by the Advanced research Projects Agency of the Department of Defense under Contract No. SD-183. The views and conclusions contained In this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency of the U.S. Government.



Computer Controlled Arm

Figure 1

To describe the programming of the computer controlled arm and hand, we will begin by discussing the tasks that we are interested in performing. Arm programming may be divided into two areas, namely, planning and execution. Planning will consist of deciding what to *do* and generating a trajectory for the arm. This trajectory will include hand actions to be performed during execution. The execution will consist of servoing the arm along the trajectory and in the performance of the desi gnated hand actions.

TASKS

In order to decide what an arm can do, let us consider fi rst the arm prope r (excluding the hand), and then the hand itself.

The prime task of the arm is motion, that is, the motion necessary to move the hand from place to place. This motion is not a simple-motion; one cannot move the joints of the arm independently i rom one value to another. Consider for example the case of turning over a block: we will obtain two arm configuration such that if the arm assumes the first configuration it will place the hand in a posi ti on where it can grasp the block; if it then moves to the iinal configuration and releases the block, the block will have been turned over as rcqui red. During the execution of this task, at *a* point after the hand has grasped the hiock , should the arm move the hand directly to its final configura tion it will in all likelihood drive the hand and block through the support (in our case the table). To avoid this we need to specify a trajectory such that the block will be lilted (in fact, such that the block is lifted and put down vertically) and, further, such that the hand and block are not driven through the support , or through any other objects during the motion. To achieve this, we will need to specify a set of relationships between joint vari ables. We must also control the acceleration of each joint if a smooth motion is to be achieved and thus the arm motion should be specified as a time-dependent coordinated motion. The hand's actions are opening, closing, and grasping actions, simple tasks requiring no servoing.

The arm can also perform actions of the form of exerting a force, where the arm motion is constrained by the hand. Turning a crank is an example of such a motion. First the hand is brought to the crank, followed by the hand action of the hand grasping the crank and then the action if the arm exerting a force on the crank in the appropriate direction.

Arm motion will be specified by a trajectory, giving joint variables as functions of time in the form of a table of values, hand action will be specified by procedures which can be executed at key points in the trajectory. Consider the task of picking up a block, a classic computer task: we have the motion of the arm to bring the hand over the block, followed by the hand action of grasping, followed by the motion of the arm in lifting the block. In such a task, sequential execution would be required, but in a task of throwing an object we would need to open the hand while the arm was in motion. Thus we have two types of hand action procedures, those which require execution while the arm is in motion and those which are executed when the arm has completed a motion. With the execution so specified, we can perform tasks such as manipulating stationary objects, turning a crank, and throwing things. There remains one further class of t asks, that of catching or intercepting other oblects. To handle such tasks we will use the following method: at the proper time a pre-computed trajectory which would put the arm into the general vicinity of the object with the correct velocity would be executed; then, as the hand neared the object, the trajectory would be perturbed by having the contents *of* an array of external cells gradually added to the set point. This,then,is the state of the execution part. One key point should be noted: in all tasks the trajectory is known ahead of time and thus the arm configuration and velocity are also known; this information is used in the section which deals with servoing the arm.

PLANNING

Let us consider the planning phase, that is, the set of programs which generate the trajectory for the execution phase. In order to cover most *of* the details we will discuss what Is perhaps the main arm task: to move the arm from its initial position to pick up an object, then to move the objet t to a final posi tion, piacing it there with a given orientation, and finally to move the arm clear.

Perhaps the key procedure will be one that, given initial and final arm configurations, will generate a trajectory between these two positions. But before we can call this procedure we must determine the initial and final positions. In the problem we are considering, we have three parts to the trajectory: the trajectory from its present position to the point at which it will pick up the block, the trajectory between picking up and placing the block and, finally, the t rajectory to the rest position.

We know the Initial position, which is the arm^{f}s current position, and we may assume that we know the rest position. The pick-up and put-down positions are undefined except tor the requirements that the hand be able to grasp the block in the first position and release the block in the second position with the required position and orientation. To specify an arm position we give the cartesian coordinates of the center of mass of the object to be picked up. We then give the orientation vector which points from one finger tip to the other. Finally, we generate a reference direction formed by crossing a vertical vector with the orientati on vector, and specify the approach angle between the fingers and this rei erence di recti on measured about the orientation vector.

For planar or convex surfaces an object may be picked up by two parallel faces on an axis containing the center of mass. This will prevent the object from rotating. One but not both surfaces may be replaced by an apex of the body. Both surfaces may be replaced by edges if they are perpendicular to the mass axis. These consi derat i ons define the set of orient at i on vectors.

To find systematically all the possible orientation vectors we first find ail vectors from the center of mass of the object that intersect and are 1) normal to any edge, or 2) normai t o any piane, 3) pass through any apex. We then search this list for anti-parallel vectors, being careful not to take both vectors from the thi rd class.

These orientation vectors are stored with the prototype, which is a global description of each object [5].

To find out how a particular instance of a given prototype may be pieked up, we simply transform the set of orientation vectors of the prototype to their space position. From this set we can Immedi ately delete any orientation vectors which require that a finger be placed on the supporting face or on any edge or vertex of the supporting face. We then have the set of possible orientations by which the object may be picked up.

The next problem Is to find out whether the arm can reach the object and to establi sh "for each possible orientation vector the range of the approach angle through which it can be reached. This is normally a difficult problem and requires the arm design to have certain properties if the solution Is to be simple. In our case the fact that the last three joint axes intersect makes the problem relatively simple [4] and we can find the ranges of approach angle, having specified the x, y, z position of the hand and the orientation vector.

Having found the ranges of approach angle for each orientation vector, we need only look at the other objects to see that the solution we have obtained does not cause an intersection between the arm and any other object.

For any body which is within reach of the arm there are normally a fair number of possible ranges of approach and orientation by which it may be picked up. Of course, if the object has to be put down somewhere after it has been picked up, then the solution is the intersection of the possible solutions at both the pick-up and put-down positions. Thus "pick up the block!" is not really a primitive operation unless very little manipulation of the objects is required; rather the primitive operation is "move the block!" For unless we consider how the block is to be placed prior to picking it up, we may find that having picked up the block we have to put it down again and pick it up differently so that we may put it down as required.

Thus having found the intersection of the possible orientation vectors at both the pick-up and put-down positions and then having formed the intersection of the approach angle ranges we may select any approach angle within the range.

These considerations of how a block may be picked up specify the missing arm configurations and we are able to call the trajectory generating procedure.

TRAJECTORY

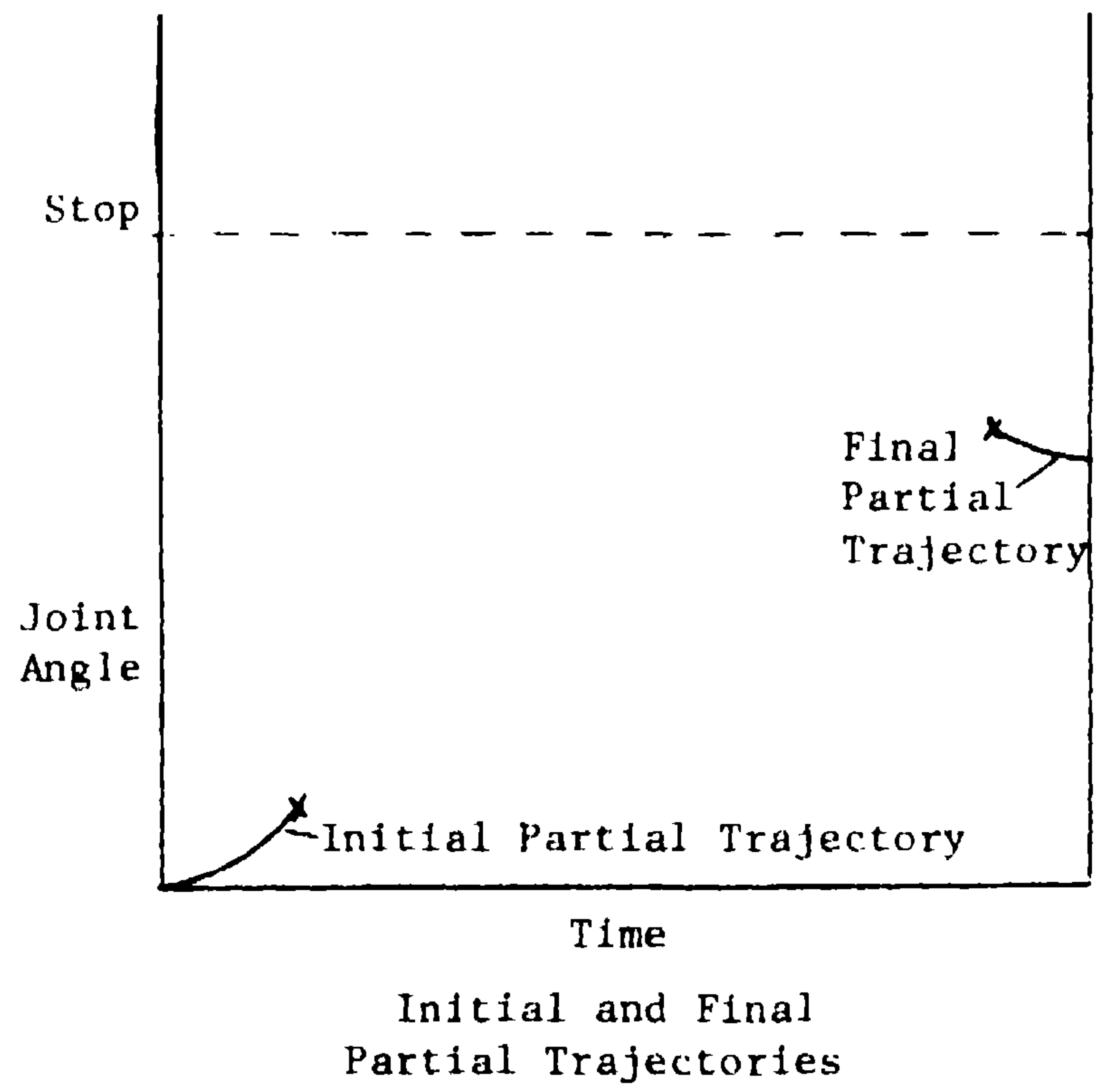
In planning a trajectory there are conflicting requirements and scarce resources. From the point of view of the arm we might want to use a "bang bang" servo, i.e. to move the arm so as to minimize time, or perhaps to move each joint so as to minimize time independently. But these are unpleasant motions involving much transient behavior and movements too fast for the arm's own saiety. Better would be a trajectory without discontinuities in position, velocity and acceleration, with zero initial acceleration and with well defined maximum acceleration and velocity. Such a trajectory would result in smooth motion of the arm without excessive velocity or acceleration.

There is another way *of* looking at the arm, and that is from the point of view of the object. Although we do not want to move the object in a straight line through space, should it rise up vertically and then move the appropriate horizontal distance before descending vertically, or should it move along a smooth space trajectory? An important requirement is not colliding with other objects, but here also we have a choice: should the trajectory move from one place to the other by feeling its way around any intervening objects, or should it move up and over all the intervening objects?

There is no simple answer to these prohlesm, and we are severely limited in time with one to two seconds being considered reasonable to plan a trajectory. What we do is to start by considering the object and to move it vertically with an acceleration initially zero.

At the end of the trajectory we move it similarly but with a controlled deceleration. Aiter the initial acceleration which lifts the object clear of the table,we consider the arm and fit a smooth trajectory which matches position, velocity and acceleration between the two partial trajectories.

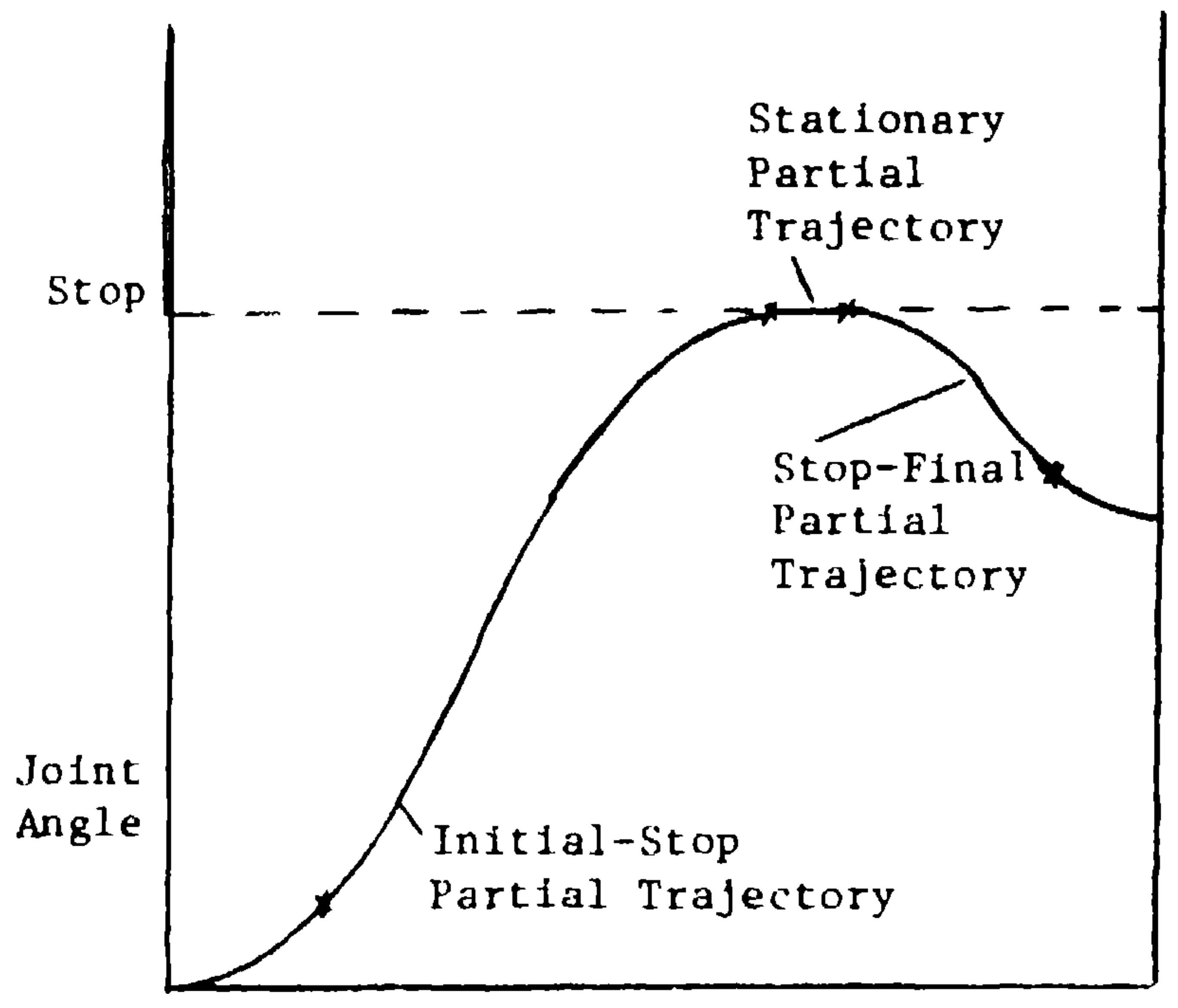
Trajectories are functionally polynomials with joint angle as the dependent variable and time as the independent variable. The trajectory for each joint is a sequence of polynomials, each specifying the trajectory for a given period of time. The total arm trajectory is a set of six such sequences. End conditions for each polynomial are continuity of position, velocity and acceleration. In the case of those polynomials specifying the beginning and end of the trajectory the initial or final velocity and acceleration are zero. To specify the Initial part of the trajectory we give a point through which the hand must pass and a time to reach this point. For picking up blocks this point would normally be about one inch above the initial position such that the block is lifted vertically, however, control of the position of this point allows a higher level planning routine to specify an initial direction which will lead the hand into clear space. The differential change in joint angles to move the hand to this point is computed, and this,together with the time, and the requirement of zero initial and final velocity and acceleration, allow us to determine the coefficients of a third degree polynomial which specifies the initial part trajectory. The final part of the trajectory is specified in a similar manner, but in this case it is the final velocity and acceleration which are zero.



Initial and Final Partial Trajectories

Figure 2

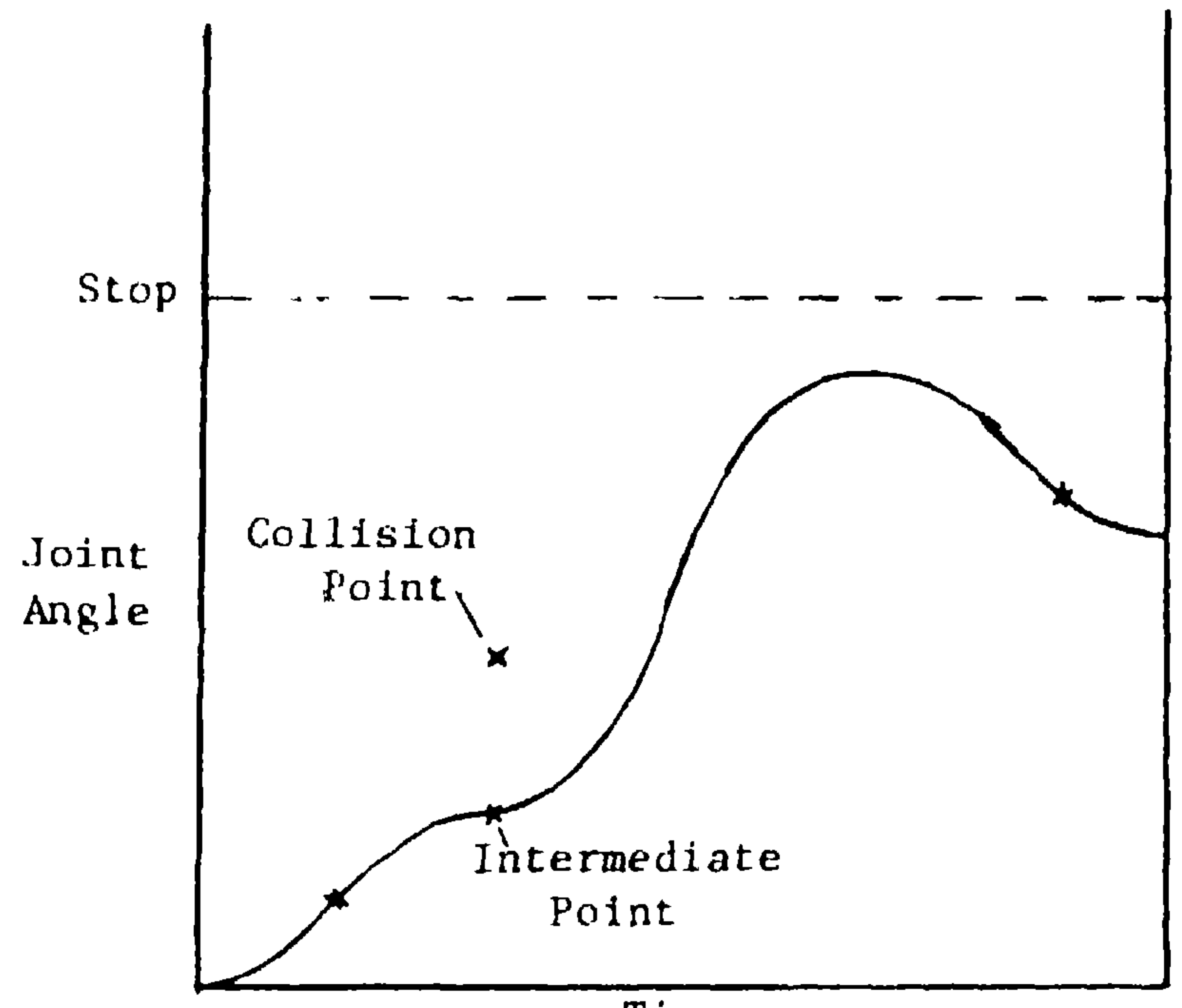
The time to move the arm from the first critical point, at the end of the beginning of the trajectory, to the second critical point, at the beginning of the end of the trajectory, is calculated on the basis of change of joint angles, and a set of partial trajectories is calculated to join the two existing pieces of trajectory. These segments of trajectories are then checked for excessive acceleration and the time increased if necessary.



Time-
Three Part Stop
Trajectory Modification
Figure 3

The trajectories are then checked for joint motion beyond the physical stops. If a joint moves beyond its physical stop the trajectory segment is broken into two parts. One part brings the arm from the first critical point to the stop, with zero final velocity and acceleration, and a second part takes the arm from the stop to the second critical point. If the sum of the times is less than the time available, then a third trajectory segment is added between these two, which simply keeps the joint stationary. If the time is greater than the time available between critical points then the other trajectories must be recalculated. Tf we find that we will collide with another object, we modify the trajectory in such a way that we pass by the object. This modification is done heuristically. Should the object with which we have collided be the table, we may simply move up, and by selecting some joint, modify the position in the upward direction. In the case of blocks this "Up" heuristic will still succeed. But in the case of towers and overhanging objects, a different heuristic will be required where we will go in and around the object. Having decided in what direction to move we compute the sensitivity to motion of the six joints and select that joint which has the greatest sensitivity in this direction. The change in angle of' this joint is then found such that the arm will no longer collide.

This change is added to the present joint angle to form an intermediate point. The trajectory is then broken into two parts: from the first critical point to the intermediate point, terminating with zero velocity and acceleration; from the intermediate point to the second critical point, starting with zero velocity and acceleration. If the time for these two trajectory segments exceeds the time that the other Joints will take, then the time must be increased and all the trajectories recalculated. If the time Is less than that available, then a stationary part will be added.



Time
Collision Avoidance
Figure 4

The modified trajectory is then checked for collision from the first critical point. If another collision is caused by this modification, then a point on the original trajectory in the vicinity oi this new collision is substituted for the first critical point, and the modification is repeated from this point. By this means, a smooth trajectory is obtained which fulfills all the requirements: vertical ascent and descent, smooth motion, and collision avoidance.

SERVOING

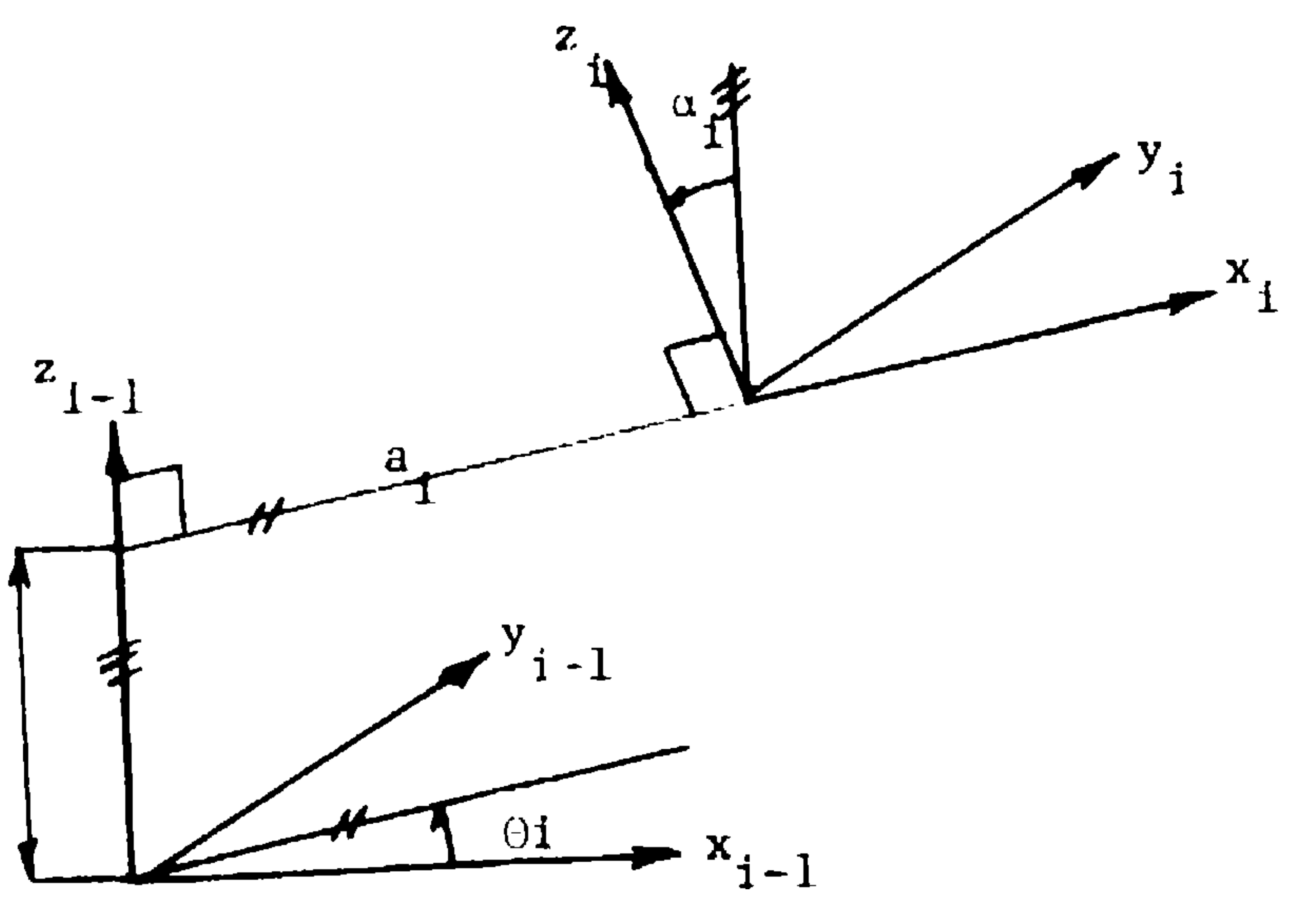
The method of servoing the arm is as follows. We have a mechanical system, "the arm," consisting of a series of links in the form of a chain, with one degree of freedom between each link. Given a table of required values of the link variables q and the present and past values of these variables', we must decide what polarity of voltage and pulse width to apply to each motor to move the arm along the required trajectory.

We can easily calculate on a joint-by-joint basis the joint acceleration necessary to keep the arm moving along the trajectory, using conventional servo techniques. This we do using both position and velocity error signals. In order to find the voltage pulse width, we substitute the accelerations so found into the Newtonian equations relating force to acceleration, to obtain the joint torques. Taking into account gear ratio and efficiency, we can then find motor torque and the related drive current. Using the back emf and motor resistance we can finally find the voltage pulse width.

In order to obtain the equations relating force and acceleration we use the matrix method developed by Uicker [6] as follows: Associated with each link is an orthogonal coordinate system fixed in the link, see Fig. . For link I the Zi axis is directed along the axis of the joint between link i and i+1. The xi axis is along the common normal between the two joint axes of the link in the direction from x_{i-1} to x . The y axis completes the right handed set.

We can relate between coordinate systems i-1 and i by performing a rotation, followed by two translations, followed by a rotation in the following manner:

1). A rotation about z_{i-1} of 0i to align x_{i-1} with xi the common normal.
2). A translation s_i along Z_{i-1} to locate the origin on the common normal.
3). A translation of a1 along x. to bring the origins into coincidence.
4). A rotation about xi of α1 to bring the z axes together.



Coordinate System

Figure 5

If we express points in link i by a column vector:

$$R_i = \begin{vmatrix} x_i \\ y_i \\ z_i \\ 1 \end{vmatrix}$$

then the relationship between coordinate systems R_i and R_{i-1} may be expressed by $R_{i-1} = |A_i| * R_i$ where $|A_i|$ is given by:

$$\begin{vmatrix} \cos\theta & -\cos\alpha\sin\theta & \sin\alpha\cos\theta & a\cos\theta \\ \sin\theta & \cos\alpha\cos\theta & -\sin\alpha\cos\theta & a\sin\theta \\ \emptyset & \sin\alpha & \cos\alpha & s \\ \emptyset & \emptyset & \emptyset & 1 \end{vmatrix}$$

if we let link Ø be the table coordinate system then we may relate from any link coordinates to RØ by

$$R_\emptyset = A_1 * A_2 * A_3 \ldots A_i * R_i$$

or $R_\emptyset = W_i * R_i$ where $W_i = A_i * A_2 * A_3 \ldots A_i$

the velocity of any point R_i may be expressed as:

$$V_i = dR_\emptyset/dt = \sum_{j=1}^{i} U_{ij} * dq_j/dt * R_i$$

where

$U_{ij} = \partial W_i/\partial q_j = A_1 * A_2 \ldots A_{j-1} * Q_j * A_j * A_{j+1} \ldots * A_i$ and depending on whether the joint is rotary or prismatic

$$Q(\theta) = \begin{vmatrix} \emptyset & -1 & \emptyset & \emptyset \\ 1 & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset \end{vmatrix} \qquad Q(s) = \begin{vmatrix} \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & 1 \\ \emptyset & \emptyset & \emptyset & \emptyset \end{vmatrix}$$

The kinetic energy of the entire system may then be expressed as:

$$K = 1/2 \sum_{i=1}^{n} \mathrm{Trace}(V_i * H_i * V^T_i)$$

with $H_i = m_i *$

$$\begin{vmatrix} 1/2(-k^2_{i11}+k^2_{i22}+k^2_{i33}) & k^2_{i12} & k^2_{i13} & \bar{x}_i \\ k^2_{i12} & 1/2(k^2_{i11}-k^2_{i22}+k^2_{i33}) & k^2_{i23} & \bar{Y}_i \\ k^2_{i13} & k^2_{i23} & 1/2(k^2_{i11}+k^2_{i22}-k^2_{i33}) & \bar{z}_i \\ \bar{x}_i & \bar{y}_i & \bar{z}_i & 1 \end{vmatrix}$$

where: k_{ijk} is the radius of gyration of link i about the j,k axes, $\bar{x}_i$, $\bar{y}_i$, $\bar{z}_i$, is the center of mass of link i. m_i is the mass of link i.

The potential energy of the system due to gravity in the -z direction, may be expressed by:

$$P = -\sum_{i=1}^{n} m_i * G * W_i * \bar{R}_i$$

where: $G = [\emptyset\ \emptyset\ g\ \emptyset]$, and g is the acceleration due to gravity. Forming the Legrangian, performing the differentiations and simplifying we obtain

$$F_i = \sum_{j=i}^{n} [\sum_{k=1}^{j} \mathrm{Trace}(U_{jk}\ H_j\ U^T_{ji}\ \ddot{q}_k)$$

$$+ \sum_{k=1}^{j} \sum_{l=1}^{j} (U_{jkl}\ H_j\ U^T_{ji}\ \dot{q}_l\ \dot{q}_k) - m_j\ G\ U_{ji}\ \bar{R}_j]$$

Although this Is the relation between force and acceleration, it is not possible to compute it in real time. At the time that we are planning the trajectory, however we know the values of q1 and we can compute the coefficients such that we have:

$$F_i = C_i + \sum_{j=1}^{n} C_{ij} * \ddot{q}_j + \sum_{j=1}^{n} \sum_{k=1}^{n} C_{ijk} * \dot{q}_j * \dot{q}_k$$

Making use of the trajectory we predict the velocities and compute the last term, combining it with the first, to obtain finally the inertial matrix.

$$F_i = C_i + \sum_{j=1}^{n} C_{ij}\ \ddot{q}_j$$

This information is then pre-computed with the trajectory and during execution the servo program, having measured q_i and calculated q , simply performs the matrix multiplication to obtain the forces. To convert from forces to pulse width is relatively simple and some frictional effects are also included.

This approach has worked well and it has proved possible to servo the arm along a trajectory resulting In a change of position of some 40 inches in less than 1 second with a final accuracy of 0.02% or 1/30 inch.

Let us see how this relates to a conventional servo. Here the output force is the weighted sum of the errors (position, velocity, etcetera), with the weighting constants usually determined experimentally. Although the acceleration may be derived as a weighted sum of the errors, the force is not proportional to the acceleration except in simple systems. In the case of the arm the term6 which correspond to the proportionality between acceleration and force, are the diagonal terms of the matrix. These terms are heavily dependent on the arm configuration which changes rapidly during motion, some of the terms varying by as much as two orders of magnitude. The off-diagonal terms of the matrix correspond to the coupling between joints, which is normally ignored. Consider, however, what happens when one of the root joints exerts a force to move the arm up: all the external joints will accelerate downwards causing a subsequent error at the next sampling period. In our case a compensating force is applied to keep all the external joints in place. The remaining terms correspond to gravity and centripetal forces; these produce constant compensating forces instead of accepting constant error offsets.

It Is believed that this work provides a systematic approach to arm programming and avoids the necessity of continually writing special programs to perform each task.

Using the computer for the servo has been central to our work and has made modification of approach simple, allowing various techniques to be tried without the necessity of building hardware.

REFERENCES

1. Scheinman, V. D., "Design of a Computer Controlled Hand", Stanford Artificial Intelligence Project Memo AI-92, Stanford University, Stanford, California.

2. Kay, A. C., et al, "The Stanford Hand-Eye Project", forthcoming paper.

3. Kahn, M. E., "Near Time Optimal Control of Open Loop Articulated Kinematic Chains", Stanford Artificial Intelligence Project Memo AI-106, Stanford University, Stanford, California.

4. Pieper, D. L, "The Kinematics of Manipulators Under Computer Control", Stanford Artificial Intelligence Project Memo, AI-72, Stanford University, Stanford, California.

5. Paul, R., Falk, G., Feldman, J. A., "The Computer Representation of Simply Described Scenes", Stanford Artificial Intelligence Project Memo AI-101, Stanford University, Stanford, California.

6. Uicker, J. J., Jr., "On the Dynamic Analysis of Spatial Linkages Using A X 4 Matrices", Ph.D. Dissertation, Northwestern University, Evanstan, Illinois, August 1965.