47.6 COMPUTER HARDWARE AND SOFTWARE ORGANIZATION

47.6.1 Software Organization

Basic Operation. The computer software is divisible into several modules in terms of function. Figure 47.14 shows the relationships of these modules. The monitor is a small program permanently stored in read-only memory (ROM) in the microcomputer. It performs basic traffic direction and boots up the supervisor. That is, when the computer is switched on, the monitor expects to find a program waiting to be read from a mass storage device. It initiates reading of this program, which is, in fact, the remainder of the permanent software. Thus, the software system "pulls itself up by its own bootstraps." This procedure is necessitated by the volatile nature of the devices used for random access memory (RAM). Random access memory is the primary memory of the computer. Information stored in RAM or ROM can be directly addressed by the computer, and so it can be accessed in minimal time. RAM is called *volatile* because the information stored in it is lost if the power is turned off. ROM is nonvolatile. That is why the monitor, at least, must be stored in ROM.

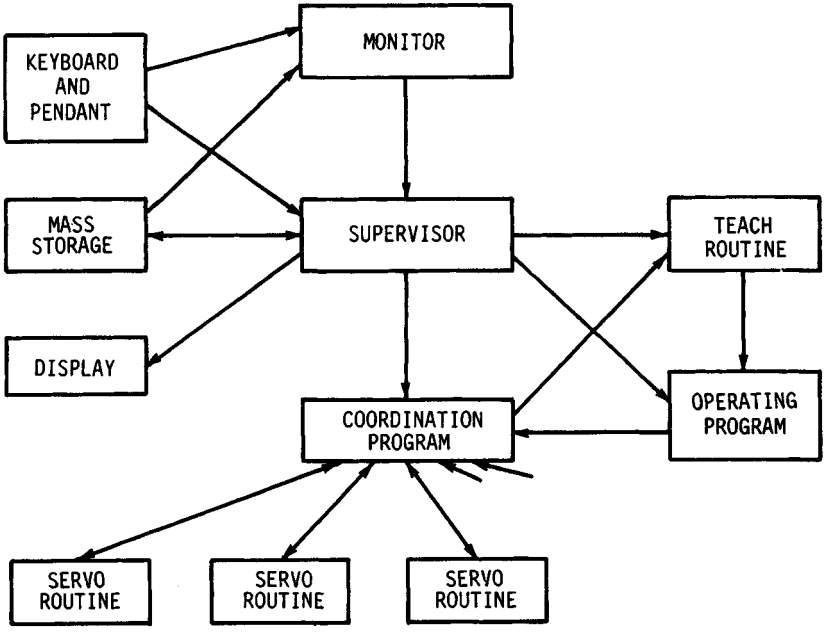


FIGURE 47.14 Computer software architecture typical of industrial robots.

In an industrial robot, storage of the entire permanent system program in ROM is a viable option. ROM is more expensive than RAM, but this configuration eliminates the bootstrap operation. There is a much more important objection to storage of the system program in ROM than cost—the lack of flexibility in terms of updating the system program. It has become customary for robot manufacturers to peri-

odically issue updated versions of the system program which provide additional capabilities. If the system program is stored on a tape cassette or disk, updating requires only substitution of the new cassette or disk for the old. If the system program is on ROM, installation of an update may be much more complicated.

Supervisory Software. Once the system software has been booted up, the direction of traffic is taken over by the supervisor, which corresponds to the operating system of a computer. The supervisor provides operator access to the computer via keyboard, function buttons, and teach pendant; displays system status; controls reading of operating programs from the mass storage device, or writing of programs into mass storage; controls the power-up and power-down sequences of the manipulator hardware; and contains modules for "teaching" (roughly corresponding to the editor of a data processing computer) and running operating programs.

Coordination Software. The coordination module is responsible for generating the position commands which are passed to the joint servos to generate smooth motions. It can be viewed as a subroutine which is called by the operating program to generate movements between programmed positions. Likewise, the coordination module is called from the teach pendant when the robot is being programmed to allow the operator to move the manipulator in a smooth and predictable way.

The coordination-level software varies greatly in complexity among different types of robots. In a simple spray-painting robot operating in joint coordinates, it may consist only of a simple interpolation routine to subdivide the differences between recorded joint positions and provide smoothly varying position commands to the joint servos. On the other hand, a robot which must provide coordinated uniform motions with respect to the fixed frame, such as those needed for seam welding, must have a complex coordination program requiring considerable computational power. The function of the coordination software is to take the data positions and the stored program instructions and generate a set of coordinated movements to implement those instructions. Since only a limited number of positions are usually stored, the computer must generate the trajectory the manipulator is to follow between each successive pair of positions.

Servo Software. The servo programs are exclusively concerned with controlling the joint actuators to produce, as nearly as possible, the joint positions commanded by the coordination module. They are, of course, present only if digital servo controllers are used.

The servo control level closes a control loop around each joint actuator. The sensor is the joint position sensor. The command signal is provided by the next higher level in the hierarchy: the coordination level. The servo loop may be configured as an analog system. This has the advantage over current digital techniques of rapid response. Alternatively, it may be a digital servo loop. Use of a digital servo allows flexible design of the servo characteristics. The capability for tailoring the response characteristics is important in manipulators in which the load and inertia seen by the joint actuator change considerably with arm position.

A digital servo loop must be serviced at frequent intervals. Thus, it is an advantage to have a small microprocessor dedicated to each joint. The microprocessor can perform the servo computations continuously without interference from the demands of other computational functions. However, it is also possible to arrange software for a single central computer which will perform servo loop computations with sufficient frequency by interrupting other computations at appropriate intervals. This arrangement avoids the problem of communication between the central computer and the servo microprocessor.

47.6.2 Microcomputers

Construction. A microcomputer is a computer which uses a microprocessor as its central processing unit. In addition to the microprocessor, the microcomputer contains the RAM and ROM memory chips and interfaces to mass storage, data entry devices, sensors, actuators, and other processors (if used). Usually it consists of one or a few printed-circuit cards on which the integrated-circuit packages are mounted. The cards are plugged into a card cage. A regulated power supply is also needed. Analog-to-digital and digital-to-analog convertors may also be mounted on cards within the computer. Actuator driver circuits, which require relatively high power levels, are usually mounted separately, often adjacent to the actuators they service.

Word Length. Compared with large computers, microcomputers are slow devices. However, the newer forms are very adequate as industrial robot controllers. Their low cost is, of course, a major consideration. The word length of a microprocessor is the number of bits which are grouped for manipulation by the processor. Microprocessors are available with 4-, 8-, 16-, and recently 32-bit word lengths. Most microcomputer controllers for industrial robots now use 16-bit processors. And 8-bit processors are useful as local processors but lack the computing power needed by the central processor. If joint position is read as an 8-bit word, the smallest change in position which can be resolved is more than one degree ($2^8 = 256$). This is quite inadequate for most robotics applications. Of course, it is possible to write software routines in assembly language or machine code to couple two words and perform double-length arithmetic. This strategy results in considerable slowing of computation.

The picture is quite different when a 16-bit word length is provided ($2^{16} = 65\,536$). Thus, computation can be carried out by using numbers with up to five significant decimal digits. At 16-bit accuracy, joint angles are resolved to within 20 seconds of arc, which is more than adequate. Thus, a 16-bit word length meets most computation needs for robotics. One type of computation for which it may not be adequate is computation of trigonometric functions, square roots, etc. As is described later, some manufacturers of recently released microprocessors have developed strategies for dealing with this problem.

Address Register. One of the most important performance parameters of a microprocessor is the length of the address register. This is what determines the number of words of RAM and ROM which can be accessed by the processor. The length of the address register is often greater than the word length used by the processor. Thus, the most commonly used 8-bit microprocessors have 16-bit address registers. A convenient unit of memory is 1000 words, abbreviated to 1K. Actually this usually means $1024 = 2^{10}$ words. A 16-bit address register permits 64K words of memory to be accessed. More accurately, 64K is $2^{16} = 65\,536$. Some modern 16-bit microprocessors have 20-bit address registers, permitting over 1 million words of memory to be directly accessed. The difference in information content between 64K eight-bit words and 2^{20} sixteen-bit words is huge. The cost of the memory modules becomes the most important factor in designing systems capable of handling this much RAM. However, the cost per bit of RAM has been falling rapidly.

47.6.3 Supporting Hardware and Software

Floating-Point Computations. There is much more to selection of a suitable microprocessor than word length, address-register length, and computing-cycle

time. Hardware and software support should also be a major consideration. If any of the robot's functions are to be performed in fixed coordinates, floating-point arithmetic, direct and inverse trigonometric functions, and square roots will certainly be required. The availability of a fast software package to provide these capabilities is an important consideration. Several manufacturers of modern 16-bit microprocessors do better than this by producing coprocessors. A *coprocessor* is an additional microprocessor chip which, when used in conjunction with the basic microprocessor, adds additional hard-wired, and therefore very fast, computational operations. Coprocessors to provide the operations listed above are available for at least two types of 16-bit microprocessors.

Interfaces and Programming Aids. Hardware support also includes interface hardware and microcomputer boards with different configurations. If a distributed processing system is to be used, the availability of a family of microprocessors with compatible characteristics and of shared bus hardware becomes important.

Another class of supporting hardware and software aids in the development of programs. Microcomputers can be programmed directly in machine code, in assembly language, or in a high-level language. Machine code is the actual binary words in which the computer operates. Assembly language is a simple system of mnemonics which allows considerable saving in programming effort with little sacrifice in computational speed. A program called an *assembler,* which is usually part of the monitor, translates the program to machine language. A development system is a computer on which programs can be developed in high-level languages such as PASCAL or C. The development system compiles the program into machine code and then downloads it into the microcomputer's memory. A cross-compiler is a program which allows the same sort of development to be done on a large computer. Likewise, a cross-assembler is a program which allows assembly language programs to be written on a large computer, assembled, and then downloaded to the microcomputer as machine code.

The cost of software development is a large cost component for most industrial robots. For this reason, it is attractive to use development systems or cross-compilers to allow relatively easy programming in high-level languages. The loss in computational speed resulting from this is outweighed by the saving in software development cost. Also, parts of the software in which computational speed is vital can be written as assembly language or machine code macros which are called by the high-level language program. Since coordination software is often very complex, it usually makes sense to write it in a high-level language. Conversely, servo routines should usually run as fast as possible, and so they are usually written in assembly language or machine code.

47.6.4 Computer Structure Options

Architectural Requirements. The software structure described in Sec. 47.6.1 directly affects the organization of the computer hardware. The supervisor and coordination modules perform systemwide functions and are best incorporated in a single central processor. The servo modules perform local functions and may be handled either in the central processor or in relatively small local processing units. The servo loops must be serviced regularly and at a cycle frequency (20 to 40 Hz) which is high relative to the mechanical natural frequencies of the servo system. Thus, if the servo loops are closed in the central processor, a timed interrupt procedure must be implemented to service the servo loops. A computer is governed by its

internal "clock," which is really a very stable, high-frequency oscillator which produces a train of precisely spaced pulses. The timed interrupt procedure counts clock cycles and, after a set number of cycles, interrupts all other processing and initiates running of the servo programs. The interruption is handled in the computer in a manner similar to the interrupts occasioned by swapping in a timesharing computer. The address location of the next instruction of the program which is being interrupted and the contents of the accumulator and other active data registers are stored in a stack so that they may be recalled as soon as servicing of the servo loops is completed.

If local processors are used to service the joint servos, the need for the complication of the timed interrupt procedure is removed. The cost is the introduction of a problem of communication between computers. Actually this is only part of a larger problem of transmission of data between different parts of the system.

Analog-to-Digital and Digital-to-Analog Conversions. Most sensors transmit data in analog form. At some point this must be converted to a digital format to allow the computer to read it. This is done by means of an analog-to-digital converter. Conversely, the driver circuits of actuators require analog input. Thus, the digital output of the computer must be converted to an analog signal by a digital-to-analog converter. Transmission of an analog signal requires only a wire pair, often in the form of a coaxial cable. With the recent advent of relatively inexpensive and compact analog-to-digital converters, considerable flexibility is possible in selecting the appropriate place in the system to perform the conversion. However, the limited number of output lines available on a microcomputer processor board usually requires that digital-to-analog conversion be done in the computer on circuit boards designed for the purpose. The outputs from the digital-to-analog converters are amplified and transmitted as analog signals to the actuator drivers.

Data Transmission. Digital data may be transmitted in either serial or parallel form. In serial data transmission, the bits of each data word are transmitted as a train of pulses. Only a wire pair is needed to do this, but the receiving device must be equipped to identify the beginning of a word and sequentially store the bits. Consequently, serial data transmission is relatively slow.

When parallel data transmission is used, a separate wire is used for each bit of the transmitted word. Thus, to transmit a 16-bit word, a cable with 16 active wires plus a ground is needed. Actually several other wires would be used to carry traffic direction signals. If a number of parallel input channels are to be fed to a microcomputer, a problem arises because of the limited number of input lines provided on the processor board. A multiplexer must be used to sequentially read the parallel input channels and to feed the resulting words to the computer. This, of course, reduces the effective transmission rate by dividing it by the number of incoming channels. Thus, if too many parallel channels must be brought in, the advantage in transmission speed as compared to serial transmission is diminished.

For similar reasons, data output from a central processor to local servo processors must be multiplexed. An alternative technique, which provides more flexibility, is the use of a shared bus. This can be thought of as a segment of memory which can be accessed by both the central processor and one or more of the local processors. This method allows each processor to run, essentially, at its own rate. Shared bus operation requires a high level of compatibility between the processors used. It also requires some means of arbitration when two or more processors seek access to a word of memory simultaneously. If too many devices share a bus, operation is considerably slowed.

Influence of Sensor Type on System Architecture. Absolute encoders read out position data directly in parallel digital format. It is attractive to feed this directly to local servo processors. This structure largely avoids the problem of multiple parallel inputs to any one processor. Although the coordination software will require joint position values, they need not be updated at high rates. Thus, this need can be serviced by serial or shared bus links.

If absolute encoders are not used, all sensor data are transmitted in analog form. It is then attractive to use a single central processor and to carry data to and from it in analog form. This allows a comparatively simple wiring design.

47.7 CONTROLLER DESIGN

47.7.1 Subdivision

The control system of an industrial robot is conveniently divided into two levels: coordination and joint servo control. The operations performed by the two levels of the control system are quite different. Joint servo control design, which is briefly discussed in Sec. 47.7.4, is similar to classical analog or digital servo design. Coordination, in contrast, in many ways is unique to industrial robots. The methods used owe more to spatial linkage kinematic theory than to control system theory.

47.7.2 Coordination

As stated earlier, coordination algorithms are those which generate the commanded joint states as functions of time from the data stored when the robot is programmed. In the case of a computer-coordinated teleoperator, the data, rather than being stored, are input by the operator in real time by means of a manual controller. The operation of the coordination algorithms is otherwise very similar.

There are enormous differences in sophistication and complexity among the coordination programs used in different industrial robots. For some types of operation, sophisticated coordination is unnecessary. There is a major division of coordination algorithms into those which operate in *joint coordinates* and those which operate in *base,* or *world, coordinates.* Most industrial robots operate in some type of point-to-point mode. Thus, when the robot is programmed, a series of discrete positions is recorded. The coordination software interpolates motions between those positions. The positions of the robot are recorded by reading the joint position sensors. To compute the position of the hand relative to a fixed frame from the joint positions, a complicated trigonometric transformation must be calculated. In a coordination algorithm which operates in joint coordinates, a simple interpolation scheme is used to command movement of each joint between successive recorded positions without regard to the gross motion of the manipulator which is produced. A typical joint coordinate scheme would command constant angular velocity at each joint with simple linear transitions between the velocities computed before and after a given recorded position. Since little computation is involved, joint coordinate algorithms can be very fast, permitting relatively high operating speeds with simple and inexpensive control computers.

The disadvantage of joint coordinate coordination is that it gives no direct control over the motion of the hand or of the remainder of the manipulator between programmed positions. The path of the hand reference point is a complicated space

curve. This is acceptable for applications such as spot welding in which only the programmed positions are important and the path taken between those positions is unimportant. Paths which cause interference with the workpiece or other hardware are easily dealt with by programming intermediate positions that take the machine away from the object it is interfering with.

Joint coordinate coordination algorithms are often used to simulate continuous-path programming. The earliest industrial robots were mostly true continuous-path devices with analog control at both the servo and coordination levels. The positions of the joints were recorded as continuous analog signals on magnetic media. Few devices of this type are now manufactured. Rather, continuous-path operation is simulated by sampling joint positions at regular time intervals during programming. The programmer must lead the machine through the desired motions in real time. A very simple joint coordinate type of interpolation routine is used during automatic operation.

Programming of robots which operate in joint coordinates can be cumbersome. It is necessary to physically place the manipulator in each programmed position. Having the operator control each joint individually, as in a backhoe, turns out to be very clumsy and inefficient. It is better to allow the operator to position the hand directly. Some robots can be placed in a mode in which the joints are unlocked, allowing the arm to be manually manipulated into the desired positions. Others use simulacra. A simulacrum is a light structure with joint geometry identical to that of the manipulator. The joints are completely free and are fitted with position sensors. The operator manually moves the simulacrum to the desired program positions. Strategies such as a "limp" manipulator mode or a simulacrum are vital for programming continuous-path type of operations. Spray painting is a good example of this situation.

When the hand path is to be controlled relative to a fixed reference frame, very much more sophisticated coordination software is necessary. For example, seam-welding robots must generate straight-line paths with high accuracy and constant hand orientation. The type of algorithm used is called a *resolved motion rate control,* or *Jacobian decomposition,* algorithm.

47.7.3 Generation of Specified Hand Trajectories

Coordination algorithms are in use which allow a robot to interpolate a specified hand–reference point path between two taught positions [47.2]. Hand orientation is also coordinated with progress along the reference point path. The point path is usually a straight line but may be a circle or other specified curve. The hand is usually rotated about an axis of constant direction. This type of algorithm depends on repetitive use of a numerical operation called Jacobian decomposition.

An infinite number of forms of Jacobian relationship can be, and are, written in the literature. However, it is possible to write a form which has a readily understandable kinematic meaning. This form also lends itself to simplification by the use of knowledge of the geometry of the manipulator chain. If ω is the angular velocity of the hand, $\dot{\theta}_K$ is the rate of rotation about joint K, and $\mathbf{w}_K$ is a unit vector parallel to joint axis K, then

$$\omega = \sum_{K=1}^{N} \dot{\theta}_K \mathbf{w}_K \tag{47.1}$$

where N = number of joint axes, usually six. That is, each joint contributes its angular velocity vector to the angular velocity of the hand. Similarly, the velocity $\mathbf{V}$ of the reference point can be expressed as

$$\mathbf{V} = \sum_{K=1}^{N} \dot{\theta}_K \mathbf{w}_K \times \mathbf{r}_K \tag{47.2}$$

where $\mathbf{r}_K$ = any vector from joint axis K to the reference point. And $\dot{\theta}\mathbf{w}_K \times \mathbf{r}_K$ is the velocity the reference point would have if axis K alone were active. These equations may be combined into the form

$$\begin{bmatrix} \omega \\ \mathbf{V} \end{bmatrix} = [\mathbf{J}]\,[\dot{\theta}] \tag{47.3}$$

where $[\dot{\theta}] = [\dot{\theta}_1, \dot{\theta}_2, \ldots, \dot{\theta}_N]^T$. And $[\mathbf{J}]$ is called a *Jacobian* matrix. It is a $6 \times N$ matrix whose Kth column consists of the elements of $\mathbf{w}_K$ followed by the elements of $\mathbf{w}_K \times \mathbf{r}_K$.

In the above, it is assumed that the manipulator joints are rotary joints. If joint K is a sliding joint, the Kth column changes form. The first three elements become 0 (since the joint does not permit rotation). The *second* three elements are now those of $\mathbf{w}_K$.

Equation (47.3) represents six scalar equations which can be solved for the joint rates $\dot{\theta}_1, \dot{\theta}_2, \ldots, \dot{\theta}_N$. Then these are used as the commanded rates for the joint servos in order to produce a nominated angular velocity ω and reference point velocity $\mathbf{V}$ at the hand.

Since Eq. (47.3) must be solved 20 to 30 times per second to produce accurate motion, it is not solved by numerical inversion of $\mathbf{J}$. Rather, the component equations are solved in analytical form to give explicit algebraic expressions for $\dot{\theta}_1, \dot{\theta}_2, \ldots, \dot{\theta}_N$ in terms of the joint variables and the elements of ω and $\mathbf{V}$.

The simplest strategy is to compute the displacement between two reference point positions and the corresponding axis and angle of rotation of the hand and to divide by a suitable time interval to get ω and $\mathbf{V}$. The machine then attempts to displace with constant ω and $\mathbf{V}$. However, since this means that the system is attempting an infinite acceleration and deceleration at the beginning and end of the displacement, it is customary to use transition segments at the beginning and end. These might be simply constant accelerations and decelerations, or they might be more sophisticated. The design of these transitions is strikingly similar in intent and in mathematical form to the design of cam motion programs.

47.7.4 Servo Design

This is a topic for specialized texts on control system synthesis ([47.3], [47.4]). Consequently, in-depth treatment is not attempted here. Rather, we review those aspects which characterize robotic applications.

Joint servos may be implemented as either analog or digital servo loops. Analog servo controllers have the advantage of faster response. Digital servo controllers present great flexibility in tailoring the controller characteristics to the system characteristics. The choice of analog versus digital controllers is also affected by the type of joint instrumentation chosen. Instruments, such as absolute encoders, which produce their output in parallel form, are most conveniently coupled directly to dedicated microprocessor controllers. Instruments such as potentiometers, resolvers, tachometers, etc., which produce analog output, are most easily coupled to analog controllers. However, inexpensive and compact analog-to-digital converters are available, so they may also be coupled to digital controllers with ease.

From the point of view of controller synthesis, the most important feature of the system is the presence of large variations in the inertia to be moved by the actuator

as well as, in most cases, in the load. The variation is, in both cases, of the order of 10 to 1. This requires robustly stable controller characteristics.

In the interests of precision, it is attractive to instrument the joint axes with position sensors. This is done in hydraulic systems. In electric systems, however, compliance and backlash in the speed-reducing transmission create stability problems. These are avoided if the position sensor is placed on the motor shaft, but at a cost in accuracy. In many industrial robot designs, gravitational loading takes care of the backlash problem, but power train compliance remains. Deflections due to it are, at least, repeatable, provided the load is repeated.

The physical system design interacts with the design of coordination software and particularly servo design in several ways. The cycle time for updating of the rates commanded by the coordination program is primarily determined by the durations of the velocity transitions which must be tracked. These, in turn, are determined by the accelerations which the joints can achieve under worst-case conditions. The acceleration at a joint required by the software during a transition must be lower than the maximum acceleration which can be achieved. To adequately track a velocity ramp, a sampling interval of about one-tenth the duration of the ramp is needed. Thus, the coordination cycle time should be one-tenth the transition interval or less. The frequency at which the servo controller operates must, in turn, be at least as high as the coordination cycle frequency. It should also be at least 10 times the bandwidth which the actuator and controller hardware would give with an analog controller to achieve performance comparable to an analog system. Thus, a typical robot system which has 0.5-s transitions and 15-Hz analog bandwidth for the joint servos should have a coordination cycle frequency of about 20 Hz and a servo sampling frequency of at least 150 Hz. If a central processor is used to perform both coordination and servo computations, it is convenient to make the servo sampling frequency an even multiple of the coordination frequency.

47.8 GEOMETRIC DESIGN

47.8.1 Structural Subdivision

Most present industrial robot geometries have been developed by trial and error, by using the drawing board and physical models. Lately three-dimensional computer simulation has become important. A theoretical basis is being developed, but some of the concepts involved are difficult. Nevertheless, it is not hard to state some basic principles.

A useful way of thinking about a manipulator structure is to regard the first three members and joints as a *regional structure* which is responsible for transporting the hand to the desired position. The outer three members and joints form an *orientation structure* whose function is to orient the hand. The logic in this is that a rotation of the hand about a "wrist" axis produces a small displacement of the hand reference point (an imaginary point in the working area of the hand fixed with respect to the hand structure). A rotation about a "shoulder" axis produces a large displacement of the reference point. Thus, orientation movements are best performed by using outboard or wrist joints, and translation movements are best performed by using inboard joints. Further, if sliding joints are used, they must be confined to the regional structure, because sliding joints do not permit rotation and so are not appropriate for orientation movements.

It is not, in general, possible to move the reference point to a desired position by using the regional structure and then to rotate the hand to its final position about an

axis through the reference point by using the orientation structure without additional movement of the regional structure. That is, translational and rotational movements cannot, in general, be decoupled. Nevertheless, for practical industrial robot or teleoperator geometries, the coupling is weak. We shall see that there are strong geometric reasons for using geometries with these characteristics.

A useful concept for study of manipulator geometry is the *reachable workspace.* It is the space within which the hand reference point may be located. Diagrams of reachable workspace are frequently presented in the literature of industrial robot manufacturers. Figure 47.15 shows a typical example.

47.8.2 Workspace Optimization

Let us consider a manipulator with six revolute joints. Successive joint axes are assumed to be either parallel or normal to one another. For the moment, the further assumption is made that there are no offsets along the joint axes. That is, the common normals to the two joint axes in each of two successive members meet on the axis of the joint connecting those members. This type of geometry is shown in Fig. 47.16. For the moment, it is also assumed that there are no mechanical limits on the rotations of the joints. The members are numbered serially from the base member (0) to the hand (6). The joints are similarly numbered, with joint 1 connecting members 0 and 1. A fixed origin point, 0 in Fig. 47.16, is defined as lying on joint axis 1 at the foot of the common normal of axes 1 and 2. Under these assumptions, the *length* of the manipulator may be defined as the greatest distance from the origin 0 to the reference point R. It is convenient to call the length of the common normal between axis i and axis $i+1$ a_i. That is, it is the length of the common normal in member i. The length of the manipulator is then

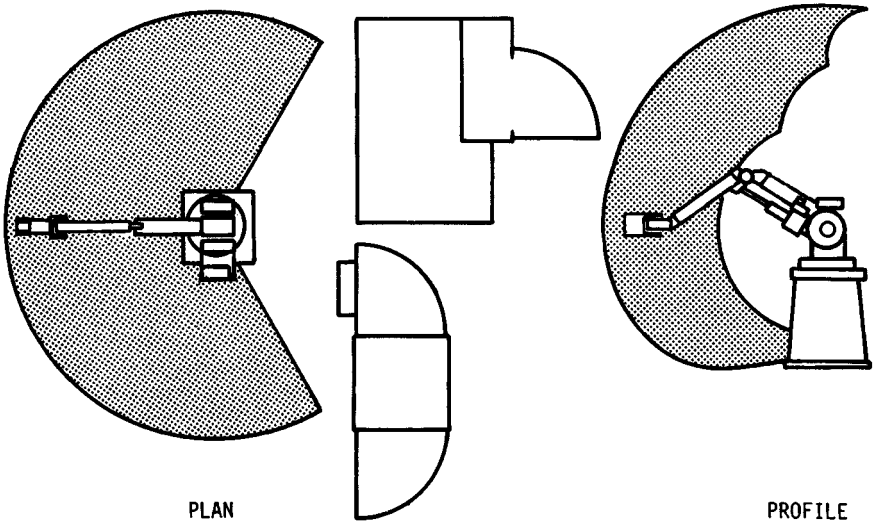


FIGURE 47.15 Profiles of reachable workspace of industrial robot. Dimensions can be determined by the grid scale, not shown, which is 6 in (150 mm). (*Cincinnati Milacron.*)

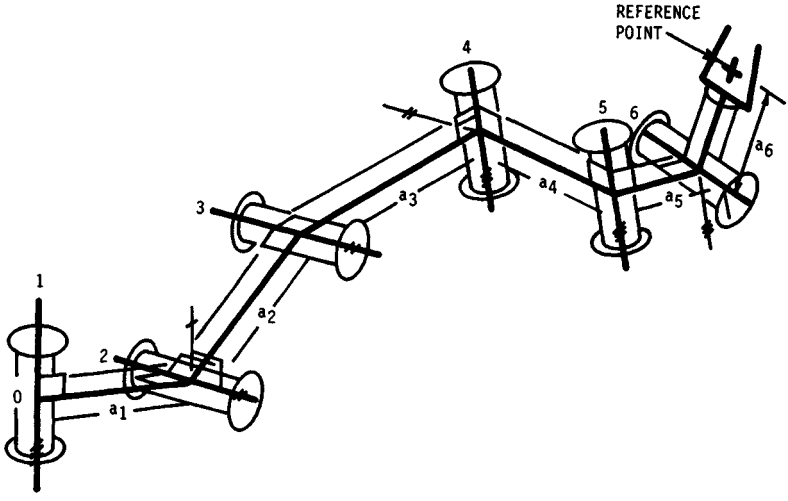


FIGURE 47.16 Manipulator chain with no offsets and with axes parallel or orthogonal. The figure illustrates the notation used in the text.

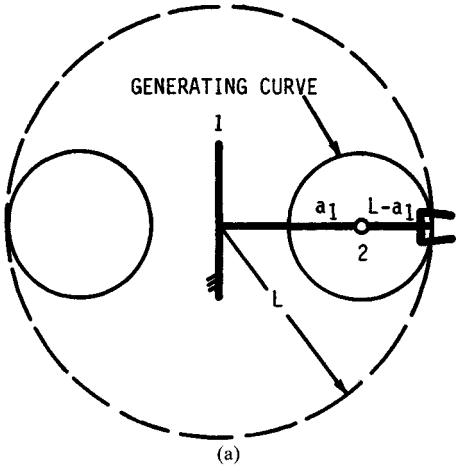
$$L = \sum_{i=1}^{6} a_i$$

The shape of the workspace boundary is governed by the inboard joints. Figure 47.17 shows the generating curves when $a_1 \neq 0$ and $a_1 = 0$. The first two joints are normal to each other in both cases. In the first case, the workspace is a torus. The generating circle radius is $L - a_1$. The center of that circle moves on a circle about axis 1 of radius a_1. When $a_1 = 0$, the workspace is a sphere of radius L. As a_1 changes, a family of tori is generated, but every member of that family fits inside the sphere of radius L. Thus, for a given length L, the maximum workspace volume is obtained when $a_1 = 0$, that is, when axes 1 and 2 intersect.

If the first two joint axes are not normal to each other but are parallel, with the third normal to the second, the situation is as shown in Fig. 47.18. The right circular torus is distorted into a toroid. Nevertheless, for a given length L, the volume is always less than that obtained when the first two axes intersect at right angles. This is, in fact, true in general for manipulator chains with six rotary joints, even when the restrictions of no joint offsets and of normal or parallel axes are removed. Likewise, the introduction of mechanical motion limits on the joints, while it substantially changes the shape of the workspace generating curve, does not invalidate this general conclusion:

> Optimum workspace volume is obtained for a manipulator with all joints rotary joints when the first two joint axes intersect each other at right angles.

The range of orientation achievable by the hand is maximized when axes 4, 5, and 6 are concurrent and when the angles between axes 4 and 5 and between axes 5 and 6 are both right angles. If these conditions are satisfied, if there are no



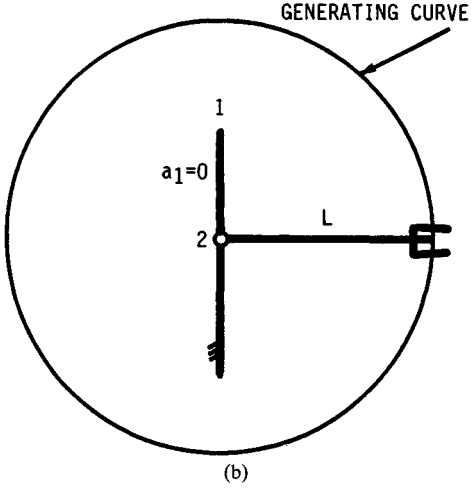


FIGURE 47.17 Generating curves when first two joint axes are orthogonal. (*a*) Axes do not intersect. (*b*) Axes do intersect; this represents the optimum for a given manipulator length.

mechanical limits on axes 4 and 6, and if axis 5 is capable of more than 180° of rotation, then the hand can be rotated completely about any line through the point of concurrency.

Since the hand reference point should be placed within the gripper area of the hand, it is usually mechanically impossible to locate it at the point of concurrency of

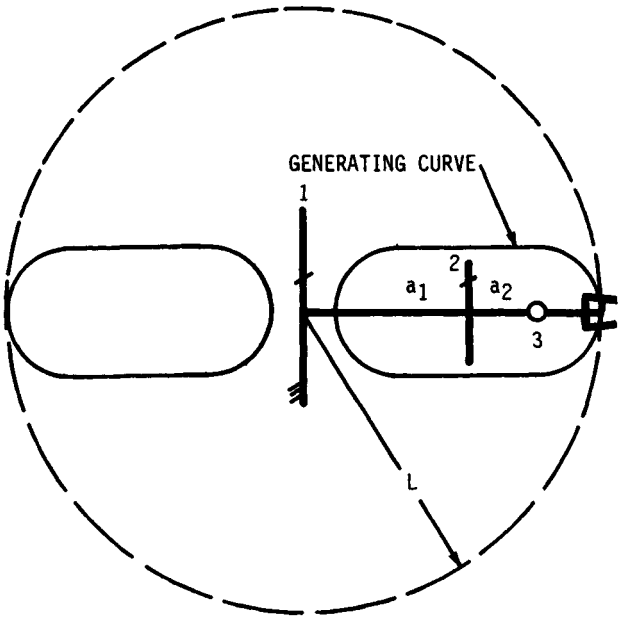


FIGURE 47.18 Generating curve when first two axes are parallel.

joint axes 4, 5, and 6 (if they are concurrent). Thus, although complete rotation about any line through the point of concurrence may be possible, complete rotation about any line through the reference point may not be. Geometric limits (as opposed to mechanical joint motion limits) on hand orientation are minimized by minimizing the distance between the reference point and the point of concurrence. That is, the hand should be as short as possible. Thus, from the point of view of dexterity, the optimum is as follows:

> Joint axes 4 and 5 should meet at right angles, and joint axis 6 should intersect axis 5 at right angles at the same point as axis 4. The hand reference point should be as close as possible to the point of concurrence of axes 4, 5, and 6. Mechanical motion limits should restrict axes 4 and 6 as little as possible. Complete rotation about these axes is desirable. Joint axis 5 should permit 180° of rotation.

If the above rules are obeyed, there is little remaining choice as far as geometric design is concerned. The placement of axis 3 with respect to axes 2 and 4 is all that remains. Axis 3 cannot pass through either the point of concurrency of axes 1 and 2 or the point of concurrency of axes 4, 5, and 6. If it does so, the chain becomes geometrically singular in all positions (the meaning of the term *geometrically singular* is explained in Sec. 47.8.3). Therefore, axis 3 either may intersect axis 2 at right angles at a point different from axis 1 or may be parallel to axis 2. Either arrangement is good from the point of view of ease of control computation. However, the former arrangement is less desirable mechanically, since it leads to large torsional moments on the arm. Likewise, axis 3 either may intersect axis 4 at right angles at a point different from the intersection of axes 4, 5, and 6 or may be parallel to it. The former arrangement is very popular, since it allows the drive for axis 4 to be conveniently

placed inside member 4. However, it leads to an awkward geometrically singular position. The latter arrangement is geometrically superior but mechanically less convenient. These two geometries are shown, with axis 3 parallel to axis 2 in both cases, in Fig. 47.19. It is no surprise that these geometries, or geometries which differ from them only in small joint offsets introduced for mechanical reasons, are extremely common among industrial robots.

Finally, it is necessary to optimize the lengths ℓ, m, p of links 2, 3, and 6 in Fig. 47.19. So far we have considered only the outer boundary of the workspace. How-

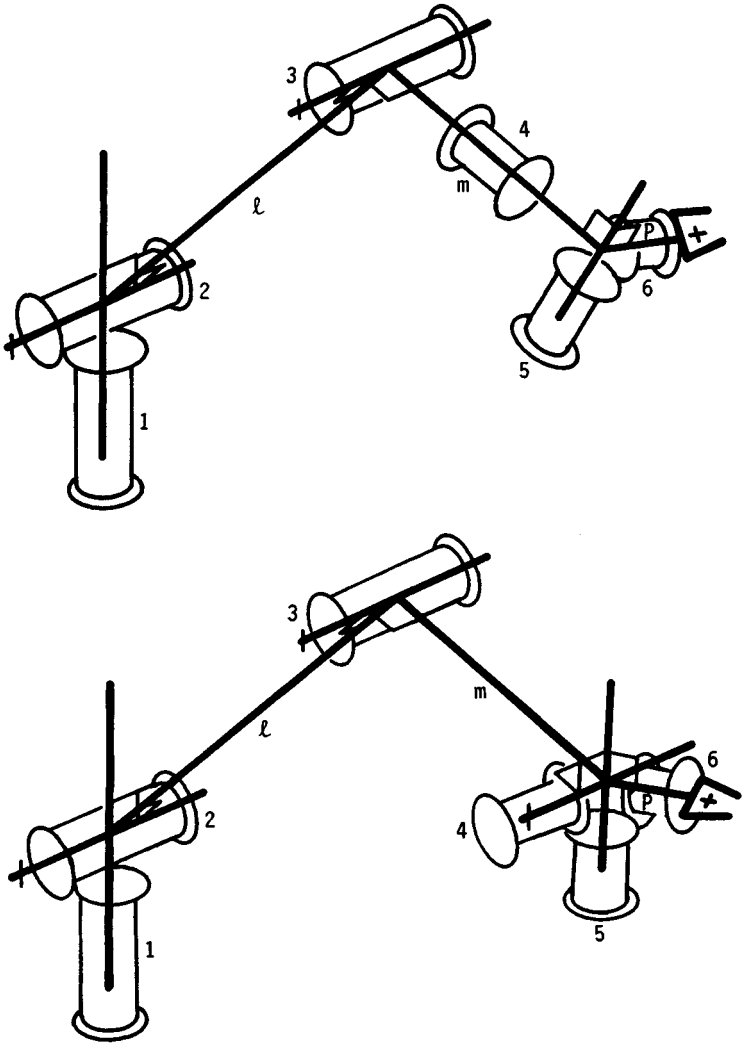


FIGURE 47.19 Two geometries which satisfy optimality conditions of regional structure.

ever, there is also an inner boundary. This is shown in Fig. 47.20 for the two cases $m + p < \ell$ and $m + p > \ell$. Clearly, the geometric optimum is $m + p = \ell$. Since there is usually structure surrounding point O, in practice a geometric capability for reaching O may not be appropriate, and a small positive value of $\ell - m - p$ may be chosen.

The above discussion applies only to geometries in which all joints are rotary joints. If one or more sliding joints are used in the regional structure, the conclusions above are no longer valid. However, a very similar approach to geometric optimization can always be taken. Sliding joints introduce one important difference. Whereas the geometric workspace for a manipulator with only rotary joints is always finite, regardless of the presence or absence of joint motion limits, that of a manipulator with sliding joints is infinite. It is only when mechanical motion limits are placed on the sliding joints that the workspace becomes finite. This is why manipulators with

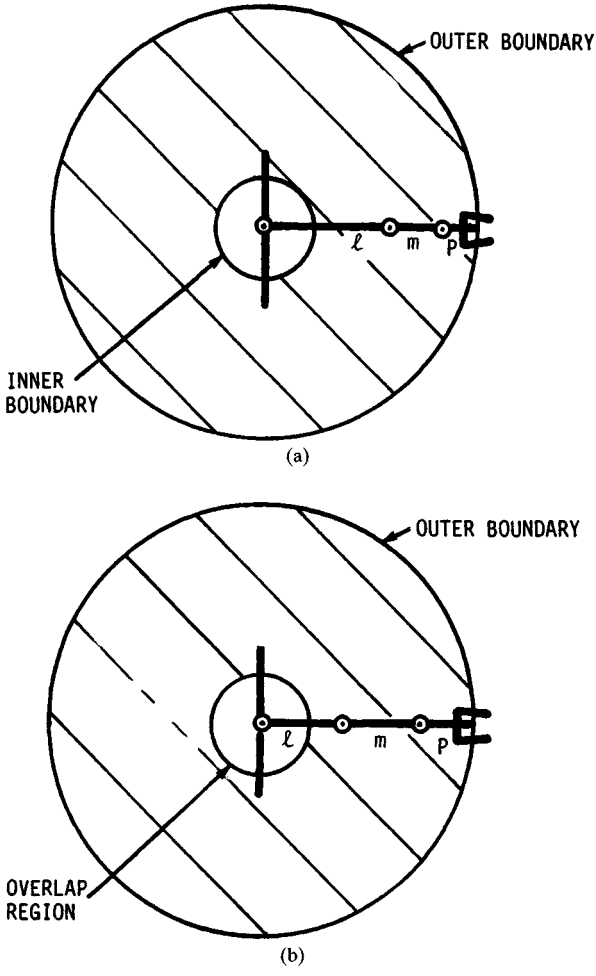


FIGURE 47.20 Relationship of inner and outer workspace boundaries to lengths of segments. (*a*) $m + p < \ell$; (*b*) $m + p > \ell$.

orthogonal slides for joints 1 and 2 or joints 1, 2, and 3 can be offered with a variety of workspace volumes. Figure 47.21 shows the workspace geometries obtained when (*a*) joint 2 is a sliding joint coaxial with joint 1; (*b*) joint 3 is a sliding joint and joints 1 and 2 intersect orthogonally; (*c*) joints 1, 2, and 3 are orthogonal sliding joints. All these arrangements can be found on industrial robots in service.

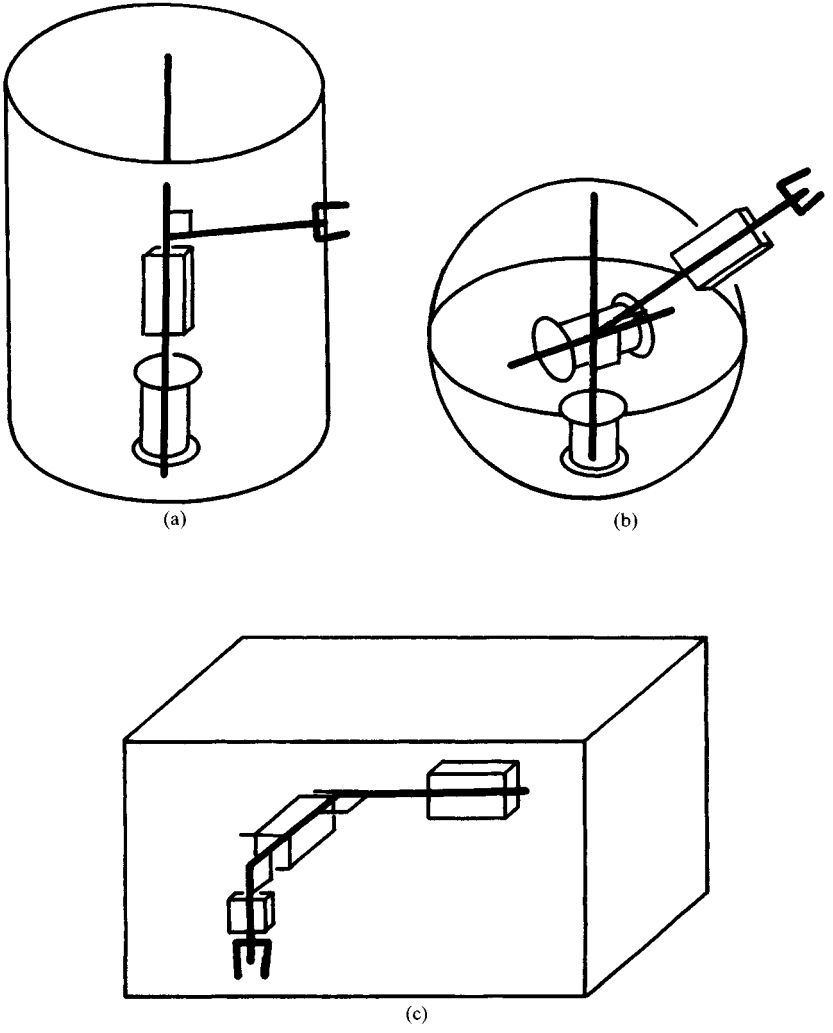


FIGURE 47.21 Some workspace geometries obtained with sliding joints. (*a*) Cylindrical geometry obtained when joint 2 is a sliding joint coaxial with rotary point 1 (or vice versa); (*b*) "spherical polar" geometry obtained when joint 3 is a sliding joint and joints 1 and 2 are orthogonally intersecting rotary joints; (*c*) rectangular geometry obtained when joints 1, 2, and 3 are orthogonal sliding joints.

47.8.3 Singular Positions

A *singular position* is a position in which a manipulator effectively loses a degree of freedom. A very simple example occurs in the roll-pitch-roll wrist configuration shown in Fig. 47.22. In the fully extended position shown in Fig. 47.22*b*, the axes of joints 4 and 6 are coincident. Thus the motions they contribute at the hand become indistinguishable. Another common example of a singular position occurs in manipulators which have three concurrent wrist joint axes. The singular position occurs if the point of concurrency is placed on the first (most inboard) axis of the manipulator. All positions in which the manipulator is fully extended are singular. This property is often used to trace workspace boundaries.

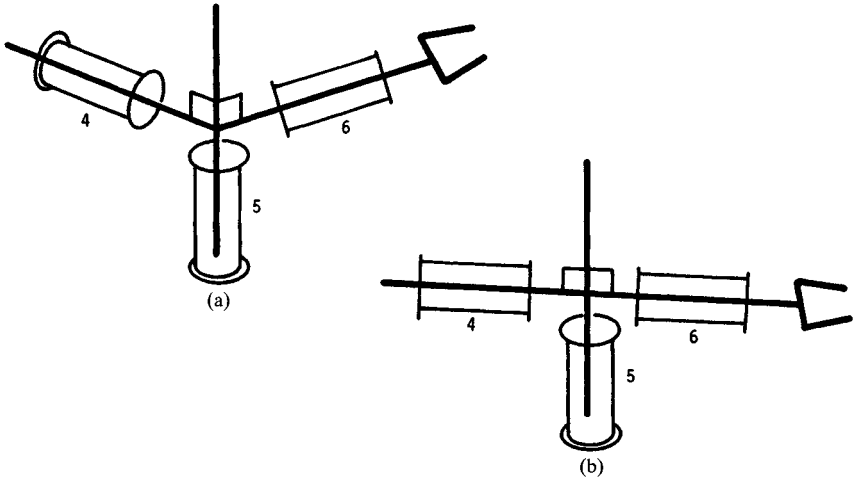


FIGURE 47.22 Geometrically singular position encountered with roll-pitch-roll wrist. (*a*) General position of the wrist; (*b*) singular position.

At a singular position, any coordination algorithm which is set up to control the hand path in *fixed* coordinates will fail. Algorithms which operate in *joint* coordinates are not affected. When the rate decomposition scheme discussed in Sec. 47.7.3 is used, the determinant of the Jacobian matrix becomes zero at a singular position.

Therefore, if coordination in the fixed frame is to be used, it is necessary to take special measures to prevent system failure at singular positions. A straightforward strategy is to provide software joint motion limits which prevent the manipulator from entering singular positions. Of course, there may also be hardware limits on joint motion. These should always be matched by limits in the software, since unexpectedly meeting a mechanical limit will usually result in program failure.

The above strategy is simple to implement and is preferred whenever feasible. It may, however, be unduly limiting. If singular positions remain which the manipulator can assume, it is necessary to make provision to detect the approach to a singular position and to provide a strategy for moving out of it. Temporary suppression of one component of the commanded hand velocity or angular velocity is an effective strategy.

One problem facing the mechanical designer of a manipulator is that of identifying singular positions to which a proposed configuration is subject. This can be done by writing the Jacobian matrix in analytical form, setting its determinant to zero, and expanding to get an algebraic condition on the joint angles which contains all the possible singular positions. This is quite feasible for the simple configurations usual in industrial robot practice, but requires a knowledge of techniques for analytical modeling of manipulator chains. The algebraic techniques involved are the same as those used in constructing coordination algorithms.

47.8.4 Accuracy and Repeatability

There are several different ways of characterizing the accuracy of an industrial robot. Each is useful, but for different purposes. *Positioning error* is the error, in world coordinates, in placing the hand reference point at a specified position. *Orientation error* is the angular error, again in world coordinates, in placing the hand in a specified orientation. *Repeatability* is the error in returning to a specified initial position after an intervening series of moves. It usually refers to the error in positioning the reference point, but angular repeatability may also be of interest. In a typical robot design, both accuracy and repeatability vary drastically over the workspace. For this reason, figures quoted by robot manufacturers can be very misleading.

There are at least four sources of error in industrial robots. One is *manufacturing error.* This includes dimensional and alignment inaccuracy and errors in the zero references of sensors. A second is *structural deflection.* This is predominantly a result of elastic deflection of the members under load. Deflections due to backlash are most conveniently regarded as components of structural deflection. A third is *servo error,* errors due to the limits of resolution of the joint position sensors or to the servo controller characteristics. A fourth is *numerical error,* errors due to the coordination algorithm. Each of these sources of error has different characteristics and affects the system accuracy in different ways.

The "teaching" process, in which a point-to-point robot is physically placed in the positions it is to assume during operation, functions as a calibration process to remove the determinate component of positioning error. The effects of manufacturing error are removed in this manner. The effects of static structural error will also be removed if the machine is "taught" with a simulated load.

Servo errors are random in nature, with range determined predominantly by the resolution of the joint position sensors. However, friction in the joint or actuation system and other effects may also have influence. Repeatability is determined by servo error. The magnitude of positioning error resulting from servo error varies strongly with position. In a six revolute arm with the first two axes intersecting, it is roughly proportional to radial distance from the intersection. In contrast, in a manipulator with orthogonal slides for the first three joints, it is independent of position for the same hand orientation. Orientation error resulting from servo error is more or less independent of position.

Numerical error is not active when the manipulator statically assumes taught positions, because no computation is involved; the system simply reads the recorded joint positions. However, numerical errors influence the accuracy with which a specified trajectory is followed and may, therefore, be significant in operations, such as seam welding, which depend on generation of precise trajectories.

47.9 TOOL DESIGN

47.9.1 Similarities to Fixed Automation

The tooling used on industrial robots ([47.5]) is very similar in character to that used for traditional, fixed automation. Jigs and fixtures, feeders, conveyors, clamping devices, and so on are all used in virtually identical form for both purposes. Although a robot is a flexible tool adaptable to varied tasks, it is usual to design the tooling which is used on it specifically for the task to be performed. Thus, if it is to be used to transfer parts, the gripping tool will be designed around the parts to be handled. This and other tooling is usually designed and constructed by specialist tool manufacturers.

47.9.2 Remote Center Compliance Devices

When a part is to be inserted into a hole during a robot assembly operation, binding due to friction usually causes failure. This can be prevented by installing a remote center compliance device in the wrist of the robot. This places the kinematic center about which the part tends to turn (under the influence of contact forces) inside the hole rather than above it. As a result, the binding problem is usually overcome.

REFERENCES

47.1 B. Roth, "Introduction to Robots," *Design and Application of Small Standardized Components—Data Book 757,* vol. 2, sec. 10, pp. 723–773, Stock Drive Products, New Hyde Park, N.Y. 1983.

47.2 R. P. Paul, *Robot Manipulators,* M.I.T. Press, Cambridge, Mass., 1981.

47.3 J. J. D'Azzo and C. H. Houpis, *Linear Control Systems Analysis and Design,* McGraw-Hill, New York, 1975.

47.4 G. F. Franklin and J. D. Powell, *Digital Control of Dynamic Systems,* Addison-Wesley, Reading, Mass., 1980.

47.5 V. Daniel Hunt, *Industrial Robotics Handbook,* Industrial Press, New York, 1983.