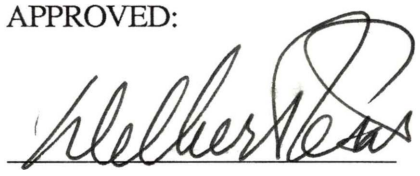
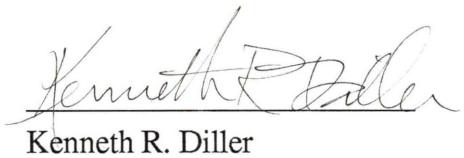
THE INTERACTIVE ASSEMBLY AND COMPUTER ANIMATION OF RECONFIGURABLE ROBOTIC SYSTEMS

APPROVED:

Supervisor: 

Delbert Tesar



Kenneth R. Diller

THE INTERACTIVE ASSEMBLY AND COMPUTER ANIMATION OF RECONFIGURABLE ROBOTIC SYSTEMS

by

RICHARD NELSON HOOPER, B. S.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

THE UNIVERSITY OF TEXAS AT AUSTIN

DECEMBER 1990

Table of Contents

Chapter 1
Introduction

Computer animation provides a means of viewing robot motion to aid in human perception and decision making for both design and operation. Current applications of computer animation to robotics have focused on the simulation and programming of existing robots. The most common uses are workcell design, off-line programming and the promotion of research programs. Since robots are usually purchased in their final configuration, there has been little demand for computer applications that aid in the design of the robot itself. The development of a modular reconfigurable robotic architecture presents an excellent opportunity to apply computer animation to the early stages of the robot design.

Interactive software packages that generate computer animations have found wide acceptance for programming and simulating industrial robots. Animated workcell design involves graphically placing the robot in its environment, also called the workcell. Machines, tools, parts and any other

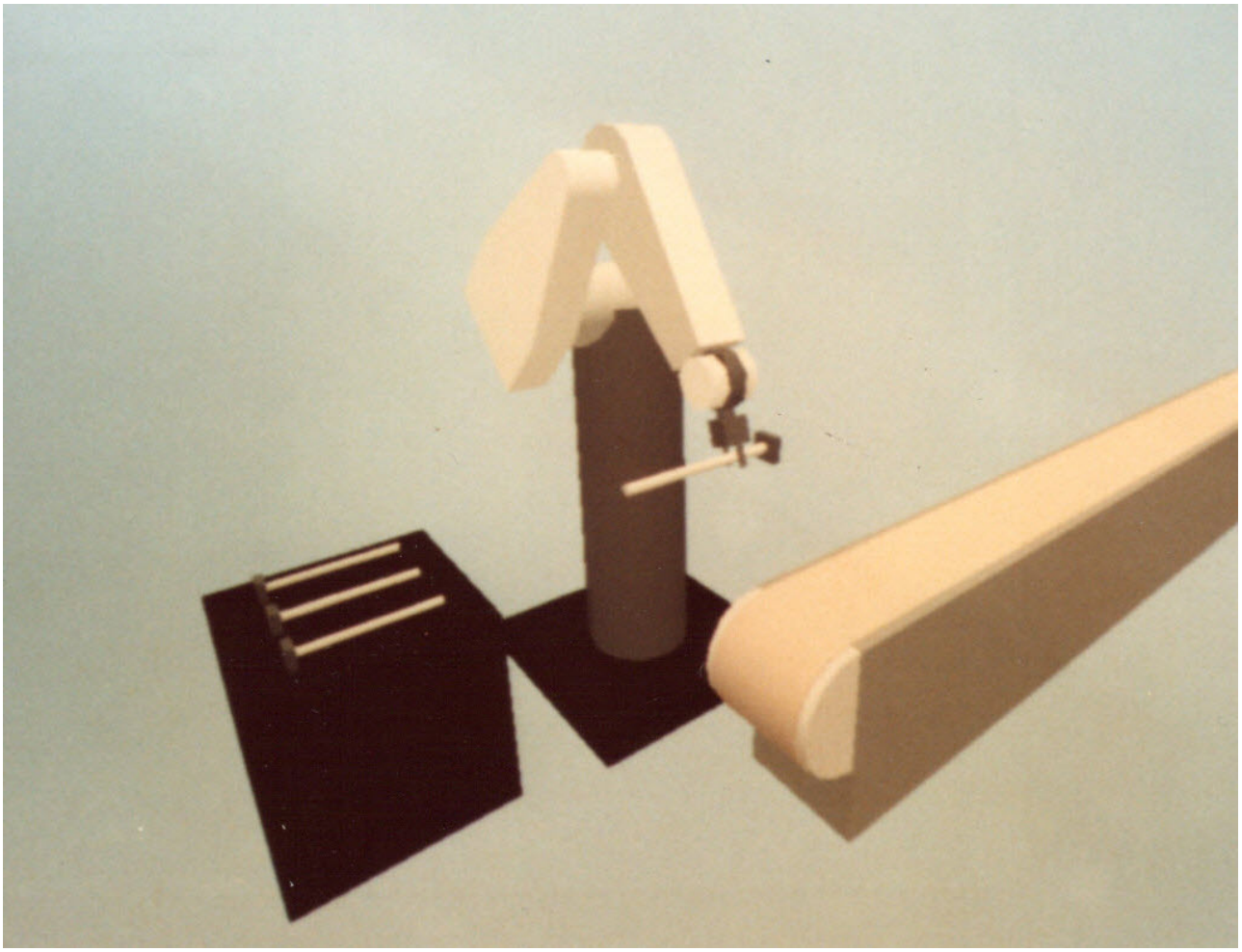


Figure 1.1: Simulation of a common industrial robot performing a pick and place operation.

objects that the robot will interact with are also placed in the workcell. Computer animation is then used to visually simulate these interactions as the robot performs its task [3]. Off-line programming uses animation to replace the actual robot while programming the robot's motion and its interactions with its environment. This allows the actual robot to remain in service while programs are being developed and thus decreases costly downtime [24]. Research program promotion is also a very important application of computer animation. Computer animation can be used to effectively convey ideas while an impressive graphical simulation of a robot performing a complex maneuver will enhance a company's or research program's high-tech image.

Current robots are purchased in their final configuration from the robot manufacturer. These robots are typically designed to perform a specific task, and if the application changes significantly the robot is rendered obsolete. The development of a generalized modular robotic architecture using a set of one, two and three degree of freedom joint modules and generic links greatly reduces the threat of obsolescence of the robotic system [46]. A modular architecture also represents an excellent opportunity to use interactive computer animation in the development of the robot itself, as well as in the development of robot technology through research enhanced by the use of three-dimensional computer animation as a visualization tool.

Available graphics workstations can produce very smooth animations of a robot moving in a complex environment. Dedicated graphics hardware performs the calculations necessary to display solid-surface models on the computer screen. These images are able to be displayed in color, three-

dimensions, perspective and with hidden surfaces removed, resulting in very realistic animations. The main drawback to using these graphics workstations is that writing the programs for computer animation is complex and quite time consuming, often requiring thousands of lines of code to produce a single animation. The difficulty and significant time investment in writing the computer animation code has led to the development of application programs that aid in the production of computer animations on graphics workstations [9][11][26][32].

The possible uses of computer animation in robotics research are many. They include the development of obstacle avoidance techniques, redundant inverse kinematics algorithms, dynamic simulations, real-time calculation, manual controllers, serial configurations, parallel configurations, mobile robots, hybrid systems, control in the small and world model databases. The use of computer animation to enhance the research on these issues has been hindered by the lack of a generalized modular and reconfigurable robotic architecture. Each robot animation must be designed one at a time and becomes obsolete when the robot design or the task changes.

1.1 Current Uses of Computer Animation in Robotics

1.1.1 Workcell Design

Computer animation enjoys widespread application in the design and programming of robotic workcells [3]. In animated workcell design the robot is graphically placed in an environment with the tools, machines and other objects that it will interact with while performing its task. The arrangement of these objects and the robot path through the environment can be visually simulated. Potential problems often become illuminated during animation and the cycle times of the robot can be studied.

A robot workcell contains all of the physical objects that the robot will interact with while performing its task. The cell may include process machinery, inspection equipment and material handling devices as well as many other types of specialized tools. Cell layout involves graphically placing the robot among these objects such that the robot may perform its task as efficiently as possible. All of the objects must be placed within the robot's reach. The use of computer animation allows many different arrangements to be simulated and compared without requiring that the actual robot be taken off line.

There is also an extremely large number of paths that the robot can take while performing its task. Interactive computer graphics has proven to be an effective method of visualizing the path planning process [51]. The workcell designer visually inspects the configuration of the cell and chooses the path for

the robot to take while performing its task. The path is then communicated to the graphics program for animation. The cell designer then visually evaluates the path and makes iterations until acceptable results are achieved.

The computer animation will highlight potential problems with the workcell design and path plan before the actual robot has a potentially disastrous problem. An excellent example of this is collision detection. The workcell designer can see collisions between graphical objects and modify the cell layout and path plan before any actual collisions occur. It is also possible to have the computer animation automatically check for collisions and visually highlight them, such as by changing surface colors. The consequences of a collision in the actual workcell could be extremely expensive as well as creating a tremendous safety hazard.

Cycle times are another aspect of workcell design that can be studied using computer animation. The cycle time refers to the amount of time that it takes the robot to perform or cycle through its task. The cycle time can be predicted by incorporating a dynamic model with the computer animation. Cycle time is a very important consideration when integrating the robot into the entire production process. Computer animation helps the workcell designer envision the phase relationships between the operation of the robot and other processes involved in the production process. Accurate prediction of cycle times helps the production engineer determine what level of throughput can be expected from the robotic workcell.

1.1.2 Off-Line Programming

Off-line programming applied to robotics allows motion and object handling programs to be developed without requiring the use of the actual robot. An obvious benefit of off-line programming is that the actual robot can remain on line while these programs are being developed, thus reducing downtime. Another benefit of off-line programming is that no human is required to accompany the robot into its environment to lead the robot through a teaching routine. Precision may also be improved through the use of off-line programming. Interactive computer animation allows the robot's path and object handling instructions to be generated and then visually evaluated for performance.

Traditional robot pendant programming requires that the robot be removed from service and that a human teacher lead the robot through its motions. The robot keeps a record of the movements of each joint and then merely cycles through the movements each time it performs its task. If the task is complex or requires a high degree of precision, many iterations must be made until an acceptable program is developed. Manually programming a robot like this may take as much as three hundred hours and must be done each time the robot is presented with a new task [35]. Taking the robot out of service for this amount of time adds greatly to the cost of using the robot to perform a production task.

Many of the current applications that seem to be ideally suited for advanced robotics are in environments that are hazardous to humans. This

precludes the use of traditional pendant programming techniques that require a human to accompany the robot into its environment. Two excellent examples of promising applications for robots in environments that are hazardous to humans are in outer space and in radioactive environments. Besides the fact that these environments are hazardous to humans, space and nuclear applications are also extremely sensitive to errors. An unintended collision between the robot and something in its environment could endanger human lives and cost millions of dollars. Previewing the robot's motion with computer animation could allow the detection and correction of these errors before they actually occur and cause damage.

In traditional robot pendant teaching the robot receives its motion instructions manually as the human moves the robot through its task. During off-line programming the robot receives its motion instructions from a human via a computer terminal. Computer languages that can efficiently communicate these motion instructions to the robot have been developed [32]. These languages also incorporate commands for many common robot tasks, such as object handling, robot motion and pick and place operations. Interactive computer animation is another alternative for communicating the human's instructions to the robot. Computer animation can allow the robot to be manipulated in joint space and the results visually displayed or, if the inverse kinematics are available, the robot can be graphically led through a visual simulation of pendant teaching on the computer terminal. Keyframing is a method for doing this graphical pendant teaching. Keyframing techniques allow the robot programmer to enter several key positions of the robot along the

robot's path. Splines or other curve fitting techniques are then used to interpolate a smooth path between the points.

1.1.3 Solid Modelling

Solid modelling techniques may be used to apply computer animation to robotics. The objects created in solid modelling are represented as filled volumes rather than simply as surfaces. These filled volumes obey Boolean set operations such as union and subtraction [1]. Interactive computer animation can be used to graphically construct a complex robotic structure. Obstacle avoidance issues may also be addressed using solid modelling. Sophisticated solid modelling programs have the potential for automatically generating dynamic system parameters, such as the mass content and compliance of the components of the system.

Solid modelling programs are commonly based upon a small set of modelling primitives upon which Boolean set operations may be performed. Typical primitives include cylinder, cone, box, prism, sphere and toroid. Interactive computer animation allows these primitives to be manipulated on the screen to build larger structures. Boolean set operations as well as geometrical relationships are used to define the topology of the system. For example, a cylinder might be subtracted from a box to provide a through-hole for a bolt or screw.

Solid modelling may also find application in obstacle avoidance. The solid modelling algorithms for manipulating volumes in space can be applied to

obstacle avoidance using potential fields. The potential fields method of obstacle avoidance places imaginary fields around the robot and obstacles in the environment. These fields can then be used by a computer algorithm to find a clear path between obstacles. Under teleoperation control the force fields can be used to provide force feedback to the handgrip held by the operator in order to help the operator steer around obstacles. Computer animation may be used to display these potential fields.

A sufficiently complete and accurate solid modelling program might also be used to generate dynamic system parameters automatically from the solid model. The mass content of the system would seem to be one of the most available system parameters. Stress and strain relationships might also be incorporated into the model. The stress/strain and mass content parameters could be determined by including material properties as features within the modules. Eventually a complete dynamic model might be automatically generated from the solid model. Computer animation could then be used to display a dynamic simulation that receives joint torques and external forces as inputs. It should be emphasized that a robot is a very complex structure with many components. Non-linearities associated with electro-mechanical and hydraulic actuators would seem to be very difficult for current solid modelling programs to address.

1.1.4 Promotion

The promotional aspect of computer animation is often overlooked when discussing the application of computer animation to robotics. Projects and ideas can be illustrated visually. Feasibility and proof of concept can be studied before any hardware is constructed. High quality computer animation also presents a progressive and high-tech image for the company or research program.

Computer animation is an excellent method of illustrating projects and ideas. Complex and difficult topics can be demonstrated in a visual form that is easily understood. This can be especially useful if the audience is not technically oriented or is only superficially acquainted with the topic. The animation can be used as a simulated robot when the actual robot is not available at the demonstration site. The animation may also be used to show progress on a project during interim inspections. Data from feasibility and proof of concept studies can also be presented using computer animation. These studies are often done before actual hardware is available. The visual presentation of a robot performing a difficult maneuver is much more impressive than numerical data for presenting results.

High resolution videotapes of complex scenes can be created one frame at a time and then played back at the speed that the robot will actually perform the task. These tapes can be shown at locations where neither the actual robot nor a graphics computer is available.

Chapter 2
Modularity

Modular design is the construction of large systems from smaller discrete units. The benefits to approaching the design of robotic systems from a modular perspective are numerous. From a finite set of one, two and three degree of freedom joint modules and generic links, a general robotic architecture can be developed. The modular robot would be reconfigurable to perform varied tasks and thus reduce obsolescence [46]. The design of the robot is simplified and cost can be reduced by using the same modules in a wide variety of configurations. The integration of technology is facilitated because a new robot does not have to be built in order to test a new idea. The cost of sending payload into space is tremendously expensive, thus making very attractive a robot that could perform many different tasks. Every robot that is sent into a nuclear environment immediately becomes contaminated and must be decontaminated or disposed of. This reality also makes attractive the use of a robot that can be reconfigured to perform many tasks.

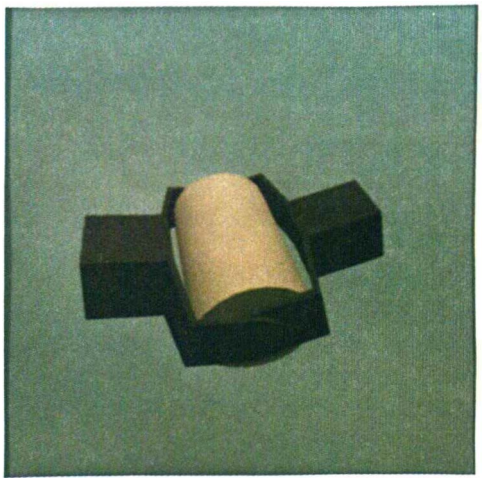


Figure 1.2: One DOF revolute

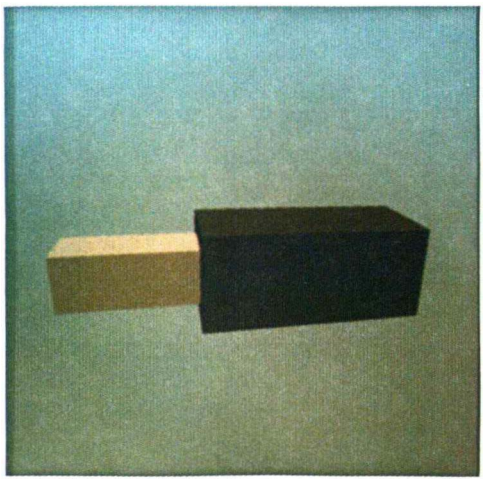


Figure 1.3: One DOF slider

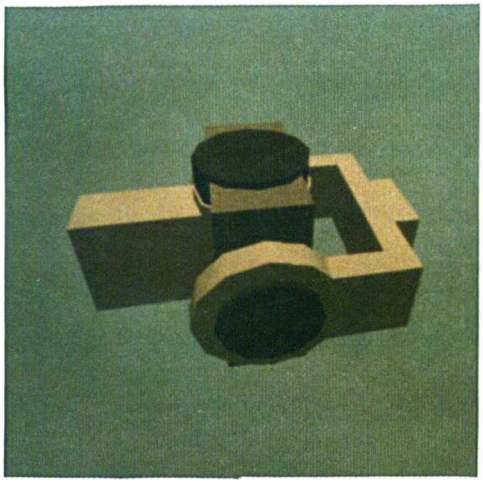


Figure 1.4: Two DOF knuckle

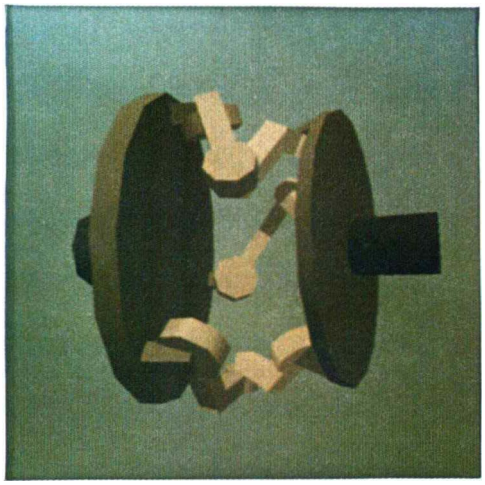


Figure 1.5: Three DOF shoulder

There are numerous issues involved in the development of a modular and reconfigurable robotic system. The intelligence and decision making system that will deal with the inverse kinematics of a reconfigurable and possibly redundant system must be developed. A controller that can adapt to reconfiguration of the system under control must also be designed and built. Indeed modularity must be incorporated into every aspect of the robot's design in order to have a truly modular and reconfigurable system.

The creation of a computer animation system for modular and reconfigurable robots does not require that many of these issues be resolved. This has allowed the development of the modular and reconfigurable animation system to lead the development of the actual physical system by several years. The computer animation can now be used as a tool in the development of the technology necessary to realize the creation of a truly modular and reconfigurable robotic system.

A modular approach to robot design also provides many direct benefits to the computer animation of robot systems. The modular computer animation package builds graphical robots that are inherently reconfigurable to animate many different robot configurations performing a wide variety of tasks. The creation of each animation is simplified to the extent that no graphics experience at all is necessary to quickly create high quality animations. The performance of the computer animation is also increased because modularity allows many calculations to be done prior to beginning the actual animation sequence.

2.1 Benefits to Computer Graphics

2.1.1 Reduces Obsolescence

Obsolescence is an issue that should be addressed whenever making the decision to buy equipment. Obsolescence should also be considered when making the decision to buy or write computer software. An animation package that seems fine because it has a large library of currently available robots may not be acceptable when the next generation of robots become available but are not in the library. The modular approach incorporated into the computer animation does not limit the computer animation to the current level of robot sophistication.

Current computer animation systems for robotics typically include a library of commonly available industrial robots. This has limited the application of computer animation to the current level of robot sophistication. A new model must be built every time a new robot design is to be simulated. Typically these animations are complex and require the work of a person that has experience with computer graphics. This method of producing computer animations is similar to the one-off manufacturing common in robotics [46]. One-off manufacturing is the design and production of expensive robots that are capable of performing only one class of task. As the task changes, the robot becomes obsolete. The same principle may be applied to the current practice of providing libraries of robot animations with computer animation packages for robotics.

As new robot technology is developed and applied, the animation library becomes obsolete.

2.1.2 Simplifies Creation of Animation

Modularity greatly simplifies the creation of computer animation for robotic systems. A modular computer animation may be quickly constructed from objects that the robotics engineer is familiar with, such as links and joints. The modular approach to computer animation allows increasingly complex systems to be graphically constructed and easily used by the design engineers involved in the development of new robotic systems and technologies. This allows computer animation to be used by the designer of robotic systems in the early stages of robot development.

The current state of the art in computer graphics workstations can display very smooth three-dimensional surface animations with a lighting model, perspective and hidden surface removal. Unfortunately, generating each animation could easily require thousands of lines of computer code. This code must be written by someone who is experienced with computer graphics and adds significantly to the cost of using computer animation. Software packages that aid in the development of computer animations are available. These software packages typically include graphical modelling primitives such as box, wedge, tube, cone and sphere, as well as transformation operations that are used to assemble the primitives into more complex models. It also must be specified where each degree of freedom is and what types of motion are

possible at each joint. Although creating animations in this manner is easier than writing computer code, it can still be quite time consuming and difficult for someone who is not experienced with computer graphics. The adoption of a generalized and modular robotic architecture greatly simplifies the creation of computer animations for robotics applications. The animation is created by assembling scalable one, two and three degree of freedom joint modules and generic links into either a serial, parallel, mobile or hybrid robot configuration. The use of graphics primitives and graphics commands is not required. An animation that might require hundreds of hours writing complicated graphics code is reduced to a few minutes interactively assembling the robot from available joints and links in a modular and reconfigurable animation programming environment.

The time required to generate computer animations of robotic systems has led to the development of software applications that come with a library of available robot animations. This limits the use of computer animation to the simulation of existing robots and has hindered the application of computer animation to the early stages of robot development. A modular and reconfigurable architecture allows the designer of robotic systems to easily generate computer animations without the need for any computer graphics experience. An extremely large class of robotic systems can be animated and the motion visually evaluated by the robot designer. The animation of these designs allows experimentation with inverse kinematics routines, manual controllers and decision making schemes. The modular reconfigurable

computer animation system provides the robotics engineer with the equivalent of an animated robotic testbed.

2.1.3 Increases Performance

The modular architecture facilitates an increase in the performance of the computer animation. This is because a finite number of modules are used to create a large class of robotic systems. The algorithms that define the geometry, surface properties and forward kinematics of each module are known. This allows many calculations to be performed and stored in fast access memory before beginning the actual animation sequence.

The graphics terminal displays a surface representation of the robot. Mathematical algorithms may be used to define points representing vertices of polygons that lie on the surface of the robot. This type of surface description is called polygonal representation. A lighting model also requires that the surface normal at each vertex be calculated. An animation of a typical robot scene can require thousands of polygons. Since calculating these surface points is done in software and often requires transcendental functions, it can be seen that these calculations represent a significant overhead if they must be performed during the animation sequence. The modular robot is created from a finite set of existing modules. This allows the calculations that are necessary to generate the surface description of the robot to be performed as soon as the modules are scaled and added to the graphical robot model. The results can be stored and the calculations do not need to be repeated during the animation sequence.

The performance of the forward kinematics is also improved by the modular architecture. Each module is drawn at the current position and orientation of the local coordinate frame; the frame is then left in place for the next module. The actual translation and rotation commands which represent the joint variables are contained within each module. Translate and rotate commands, which may be implemented in hardware for significant performance advantage, can be used to move the coordinate frame and perform the translations and rotations at the joints, thus eliminating the need to perform the forward kinematics explicitly in software.

2.2 Benefits to Robotics

The benefits to robotics of a modular and reconfigurable architecture are many. The benefits stem from the fact that a finite set of one, two and three degree of freedom joint modules and generic links can be used to build a general mechanical architecture that represents an extremely large class of robotic systems. Robotic systems constructed from these modules will be reconfigurable to perform many different tasks thereby reducing obsolescence and lowering costs.

The design of the robotic system is also greatly simplified by the adoption of a modular architecture. This is because the design parameters can be broken down and addressed in smaller groups that are contained within each module. Input-output relations for compliance, inertia and damping can be developed for each module independently. The ability to reduce the design

parameters to a manageable set also encourages the use of feed-forward design and control.

The integration of new technology is also facilitated by adopting a generalized mechanical architecture. This is because a complete new robot does not have to be built to test the application of a new technology. Less conservative design is also possible because the design is addressed at a modular level. Direct performance comparisons are also possible by changing a single module without changing the rest of the system.

Operation in environments that are hazardous to humans provides an excellent opportunity for the use of robotic systems. Operation in space and in radioactive environments are two examples of hazardous environments that have been considered for robot application. It is, however, very difficult and expensive to place a robot in these environments. This makes a modular and reconfigurable robot that can perform varied tasks quite attractive. Because it is difficult to access a robot in space or nuclear environments, it is important that the robot be as simple to repair as possible. The repair of a modular robot can be performed by simply replacing a broken module with a working one. Fault tolerance is also a crucial issue in space and nuclear environments and can be addressed at the modular level.

2.2.1 Reconfigurable to Perform Varied Tasks

The idea that robotic systems should be reconfigurable to perform varied tasks is implicit in the generalized modular robotic architecture. In order that the robotic system be truly reconfigurable, modularity must be addressed in every aspect of the robot design. This includes the design of standardized electrical and mechanical connectors as well as the development of intelligent controllers and decision making systems that can adapt to reconfigurations of the system model. The threat of obsolescence of a modular and reconfigurable robot is reduced. The cost of producing a modular and reconfigurable robot would also be reduced in the long term.

Reduced obsolescence of production machines was seen as an early promise for robotics. Current robotics technology has failed to fulfill that promise. Today's robots are designed for a specific task, such as welding or pick and place, and the robot becomes obsolete if that task changes. A modular robot would be able to be reconfigured to adapt to changes in manufacturing demand, such as more bolt tightening and less welding. A precision task could be accomplished by using stiffer modules or perhaps the cycling time could be shortened by using lighter modules in an application that requires less precision. A modular robotic architecture would do much to fulfill the early promise of reduced obsolescence.

The development of a truly reconfigurable robot would be more expensive in the short term than producing one typical non-modular industrial robot. This is likely a contributing factor to the lack of modularity found in common industrial robots. The lack of a modular architecture can, however, be seen to be extremely expensive in the long term. The Cincinnati Milacron t3 series required seven million dollars of development expense and seven years of development time to bring to market [46]. The NASA space shuttle flight manipulator was delivered at a total cost of one hundred million dollars [46]. The next generation shuttle manipulator is expected to cost up to one billion dollars [46]. A modular mechanical architecture would significantly reduce the cost of producing robotic systems by allowing larger production runs of a finite number of modules that may be assembled into an extremely large class of robotic systems. This is analogous to the initially high cost of designing and producing microprocessors and integrated circuits that may be wired into a virtually limitless number of electronic and computer applications. This ultimately results in the enormous benefits of scale that are reflected in the price, performance and rapid advancements found in the electronics field. It is, in fact, almost impossible to imagine designing a new microprocessor for every computer application, yet robots are still designed and produced one at a time with virtually no regard to modularity or reconfigurability.

2.2.2 Improves Design Process

The design of a robot system is an extremely complex procedure in which a large group of design parameters must be considered. A six degree of freedom serial arm has eighteen geometric, fourty-two mass, thirty-six compliance and eighteen actuator parameters [46]. This is an extremely large number of design parameters and the coupling between the parameters is not intuitive or obvious. A generalized modular mechanical architecture breaks the design parameters into small manageable groups. Input/output relations can be developed for each module independently. A sufficiently complete and accurate model of the system allows the incorporation of feedforward techniques in the control of the robot.

The design of a six degree of freedom serial manipulator results in over one hundred operational parameters. Sophisticated structures incorporating redundancy, in-parallel linkages and multiple branching chains will generate far more available parameters. An intelligent and systematic design procedure would incorporate these system parameters into the design, however this number of design parameters is too large to face simultaneously. A modular mechanical architecture naturally breaks the system into functional mechanical units with a reasonable number of design parameters associated with each module.

The modular mechanical architecture allows input/output relationships to be developed for each module independently from the entire robot structure.

These input/output relations can describe dynamic phenomena such as compliance, damping and inertia that interact with the rest of the system at the mechanical and electrical connections. Engineering analysis and metrology can be used to develop the relationships that describe the dynamic behavior for each module. These input/output relations can then be used along with the configuration of the robot to determine a dynamic model for the complete system.

Typical robot control schemes use feedback from the output to the input in an attempt to cancel errors caused by system dynamics. Feedforward control implements an input algorithm that incorporates system dynamics in an attempt to cancel errors before they occur [28]. The actual performance of the feedforward control is dependent upon the completeness and accuracy of the system model. Because of the lack of modularity in current robots, accurate dynamic parameters are very difficult to determine. This impedes the use of feedforward control in the application of these robots. By adopting a modular and reconfigurable architecture, the dynamic model for any configuration would be determined by the topology of the system and the known dynamic characteristics of each module.

2.2.3 Facilitates Integration of Technology

Current robot design practice results in extremely long design-to-market cycles. The lack of a modular architecture also means that the failure of any one component within the robot will disable the entire system. This leads to

extremely conservative design that is unlikely to incorporate new less-proven technologies. It is also possible with a modular architecture to make direct performance comparisons by changing only one module and leaving the rest of the system unchanged.

The complexity of robot systems combined with the lack of a modular architecture has resulted in the extremely long design-to-market cycles characteristic in the robot industry. This is evidenced by the seven years it took Cincinnati Milacron to bring the t3 series of robots to market. This guarantees that any new technology that is integrated into the robot design is old by the time the robot comes to market. Implementation of improvements in materials, sensors, actuators or electronics will have to wait for the next robot design. A modular architecture with standardized interfaces allows the incorporation of new technology in the design of a module without requiring that the entire system be redesigned.

The high cost, both in terms of time and money, of bringing a robot system to market forces the engineer to be very conservative in the design of the robotic system. Newer and higher performance technologies must be rejected in favor of more proven technologies because the failure of a single component will disable the robot until repairs can be made and a recurring failure would jeopardize the entire robot design. A modular approach would allow less conservative design because the failure of one module would only require that the module be replaced, a repair which could be quickly performed. A recurring failure within a module would only require the redesign of that module.

A modular approach also allows direct performance comparisons between technologies. It is possible to change only one module while leaving the remainder of the system constant. In this way the effects of adding a new technology can be studied without corrupting the results by changing more than one discrete system component at a time.

2.2.4 Improves Performance in Space and Nuclear Applications

Operation in an environment that is hazardous to humans is a robotic application that would be of immediate benefit. A modular and reconfigurable robot would be of particular benefit in space and nuclear environments. It is, however, extremely expensive to send robots into these environments. The difficulty of accessing robots in these environments dictates that robots sent into space or nuclear environments must be as easy to repair as possible. These are also extremely sensitive environments where catastrophic results could occur from a robot fault. More than one billion dollars is set aside in the current space station budget for robotics and automation [45]. The first decommissioning of a commercial nuclear reactor in the United States cost one hundred million dollars and there are more than five hundred other reactors worldwide that will eventually follow [42]. This represents a tremendous opportunity for the application and further development of robot technology. A problem with sending robots into nuclear environments is that the hardware sent into the radioactive environment may become radioactive waste that must also be dealt with. The limited task capability of current robots would require that many

different robots be sent into the radioactive environment in order to perform the varied tasks that would be encountered. Generating this amount of nuclear waste combined with the cluttering of the environment with robots would diminish the benefits associated with using the robots in the first place.

Experience has shown that even the most meticulously designed systems will sometimes fail. This would seem to indicate that repairability be an important consideration when designing or applying robotic systems. Repairability is of increased importance in space and nuclear applications because it is very difficult to repair a robot in these environments and it is not feasible to remove the robot from the environment in order to execute repairs. The space and nuclear environments are also likely to impose tremendous stress upon the robotic system. The stresses may come from radioactivity, wide temperature variations and orbiting debris, as well as from many other predictable and unpredictable conditions and events. A modular robotic architecture facilitates repairs because the repairs can be addressed at a modular level. Servicing could be performed by simply replacing modules. This repair could be executed by humans, by another robot or even conceivably by the robot repairing itself. Fault tolerance is of extreme importance in critical space and nuclear applications [47]. A fault in these environments could very easily have catastrophic results. Graceful degradation may not be adequate, and catastrophic failures that might jeopardize the safety of the entire system must at all costs be avoided [47]. This level of fault tolerance can be addressed at the modular level by embedding redundant actuators within the joint modules. In the event of a failure of one actuator motor, the other actuator motors could take

over and operate beyond steady-state conditions for a finite amount of time. Redundant modules incorporated into the configuration of the system as in-parallel structures could also address fault tolerance at another level.

Chapter 3
Computer Animation Technology

Computer animation provides a means of visually simulating the motion of a robotic system. Computer graphics might be thought of as the aspect of computer animation that actually displays the image of the robot system on the computer screen. If many robot images are displayed in rapid succession with the joint variables changed by a small amount each time, the robot image on the screen will appear to be moving in a continuous fashion. In order to produce a computer graphic simulation of the robot, it is necessary to communicate to the computer a description of the robot's visible surfaces. Besides the information necessary to produce a static image of the robot, a computer animation requires that the robot's kinematics be known as well. Since the computer animation will be used as a tool to visualize the robot in motion, it is important to understand the precision with which the computer displays the image.

3.1 Graphics

The graphical capabilities of the computer are used to display the image of the robot. Three-dimensional graphical presentation requires the use of three-dimensional transformations which may be represented as matrix operations. The description of the surface properties of the robot determines the appearance of the computer animation. Often times there is more than one surface in a single line of sight. In these cases it is necessary to do hidden surface removal so that only the surfaces nearest to the viewer are displayed. Because of the many operations that must be performed on each pixel that is to be displayed, such as rotation, translation and perspective transformations, a pipelined computer architecture is efficient for computer animation.

At the heart of three dimensional graphics are transformation and projection operations. These operations may be conveniently written in matrix form. The operation to translate an object while maintaining a constant orientation may be written as:

$$\begin{bmatrix} x_{new} & y_{new} & z_{new} & 1 \end{bmatrix} = \begin{bmatrix} x_{old} & y_{old} & z_{old} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}.$$

The operation to scale an object may be written as:

$$\begin{bmatrix} x_{new} & y_{new} & z_{new} & 1 \end{bmatrix} = \begin{bmatrix} x_{old} & y_{old} & z_{old} & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The operation to rotate an object about the x axis by an angle theta may be written as:

$$[x_{new}\ y_{new}\ z_{new}\ 1] = [x_{old}\ y_{old}\ z_{old}\ 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The operation to rotate an object about the y axis by an angle theta may be written as:

$$[x_{new}\ y_{new}\ z_{new}\ 1] = [x_{old}\ y_{old}\ z_{old}\ 1]\begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The operation to rotate an object about the z axis by an angle theta may be written as:

$$[x_{new}\ y_{new}\ z_{new}\ 1] = [x_{old}\ y_{old}\ z_{old}\ 1]\begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The projection transformations determine how a three-dimensional scene is projected onto the two-dimensional computer screen. In a perspective transformation it might be imagined that a three dimensional scene is being viewed through a pane of glass. If a line were drawn from every visible point

on the scene back to the viewer's eye, and a dot were drawn on the pane of glass the exact same color as the points in the scene, then the two-dimensional drawing would appear to the viewer exactly the same as the three-dimensional scene. In order to define a scene for perspective transformation it is necessary to define where the pane of glass is; the near field, NF. It is necessary to know where the back of the scene is; the far field, FF. It is also necessary to know the field of view on the vertical, FOVV, and the field of view on the horizontal, FOVH. Using this terminology, the perspective projection may be written as:

$$[x_n\ y_n\ z_n\ 1] = [x_o\ y_o\ z_o\ 1] \begin{bmatrix} \frac{\cot\left(\frac{FOVV}{2}\right)}{FOVH} & 0 & 0 & 0 \\ 0 & \cot\left(\frac{FOVV}{2}\right) & 0 & 0 \\ 0 & 0 & \frac{-FF-NF}{FF-NF} & -1 \\ 0 & 0 & \frac{-2*FF*NF}{FF-NF} & 0 \end{bmatrix}$$

A simpler and computationally faster type of projection operation is the orthographic transformation. The orthographic projection is orthogonal and, unlike the perspective projection, objects do not appear to get larger as they get closer to the viewer. A box shaped enclosure may be utilized in order to define an orthographic projection. It is necessary to define the front, back, top, bottom, left and right boundaries of the box. The orthographic projection operation may then be written as:

$$[x_n\ y_n\ z_n\ 1] = [x_o\ y_o\ z_o\ 1]\begin{bmatrix} \frac{\cot\left(\frac{FOVV}{2}\right)}{FOVH} & 0 & 0 & 0 \\ 0 & \cot\left(\frac{FOVV}{2}\right) & 0 & 0 \\ 0 & 0 & \frac{-FF-NF}{FF-NF} & -1 \\ 0 & 0 & \frac{-2*FF*NF}{FF-NF} & 0 \end{bmatrix}$$

It is necessary to have a surface description in order to present a visual simulation of a three-dimensional object. The conditions at the surface of an object are what actually determine the appearance of the object. An obvious attribute of a surface is its color, or more correctly the color of light the surface reflects. Another attribute of a surface is its texture, rough or smooth, etc. A surface can reflect diffuse light, light that is scattered equally in all directions, and it can reflect light directionally. A surface will also reflect ambient light. Ambient light is non-directional light that has typically been scattered by reflections off of other surfaces. A surface may even emit light as well as reflect it. A totally complete description of even one surface could be very complex and detailed. In fact, it may be thought of as a modelling problem in and of itself. Animating a typical robot workcell requires that thousands of different surfaces be modelled and displayed many times each second. The result is that the calculation power necessary to do real-time animation of three-dimensional solid surfaces with a lighting model is quite large. In order to display a scene with one thousand surfaces at a rate of thirty frames per second requires that the computer display thirty thousand frames per second while performing all of the transformations and modelling that are necessary for each surface.

$$30 \frac{\text{frames}}{\text{second}} * 1000 \frac{\text{surfaces}}{\text{frame}} = 30000 \frac{\text{surfaces}}{\text{second}}$$

This type of processing power is available in current engineering workstations, and as more powerful graphics processing becomes available, higher resolution and more complex scenes may also be animated in real-time.

When viewing an animated scene, some surfaces will be hidden from view as other surfaces pass nearer to the viewer. Showing only the surfaces that the viewer would actually see is called hidden surface removal. Ray tracing is one method that may be used to perform hidden surface removal. Ray tracing techniques follow a straight line from the viewer through the scene, and only the points that the line intersects first are actually shown. Unfortunately ray tracing is computationally expensive because lines must be drawn through every point in the scene after the surfaces have been geometrically transformed. Z buffering may also be used to perform hidden surface removal. The z buffer is an array of numbers where each number is associated with a pixel on the screen. The numbers kept in the z buffer are the distances between the viewer and each point that will be drawn in the scene. As each new surface is transformed, the distance to the viewer is compared to the value in the z buffer and only the points closest to the viewer are actually drawn. Z buffering is computationally fast but memory-intensive as values must be kept for each pixel on the screen. For example, keeping a z buffer of thirty-two bit numbers for each pixel on a screen that is one thousand by one thousand pixels requires four megabytes of fast access memory.

$$32 \frac{\text{bits}}{\text{pixel}} * 1000 \text{ x pixels} * 1000 \text{ y pixels} * \frac{4 \text{ bytes}}{32 \text{ bits}} = 4 * 10^6 \text{ bytes}$$

Pipelined computer processing has proven to be of advantage in graphics computers. This is because many operations must be performed on each point before it can actually be displayed. These operations include geometric transformations and projections, lighting models, surface models and clipping. Specialized hardware that efficiently performs these operations can be arranged in a pipeline where many points can be operated upon at once, with a different operation performed on each one. A computer pipeline might be compared to an automobile assembly line. Many autos are being worked on at once, but different operations are performed at each station. At one station they may be putting on fenders while at another station they are putting on the wheels. The auto is finished after it has been through the entire assembly line.

3.2 Animation Methods

The animation effect is created by displaying a series of still images in rapid succession with the robot's joint displacements changed by a small amount in each image. The proper values for the joint displacements can be made available to the animation in many ways. An easy way to make the joint displacements available to the animation is to simply calculate them within the same program that is driving the animation. Another method which may be used creates a shared memory structure that may be accessed by both the graphics process and the process that is generating the joint angles. It is also possible to generate the joint angles on a separate computer and then pass the

Task	Polygon Peformance
Workcell Layout	$30*10^{3}$ polygons/second
Flexible Link Manipulators	$30*10^{6}$ polygons/second
Parameter Display by Surface Shading	$30*10^{6}$ polygons/second
Path Planning	$30*10^{3}$ polygons/second
Off-line Programming	$30*10^{3}$ polygons/second

Table 3.1: Required graphics performance for a typical scene animated at a rate of thirty frames per second.

joint angles to the animation computer via a network. For applications that do not require real-time operation, the joint angles may simply be written to a datafile for animation at a later time.

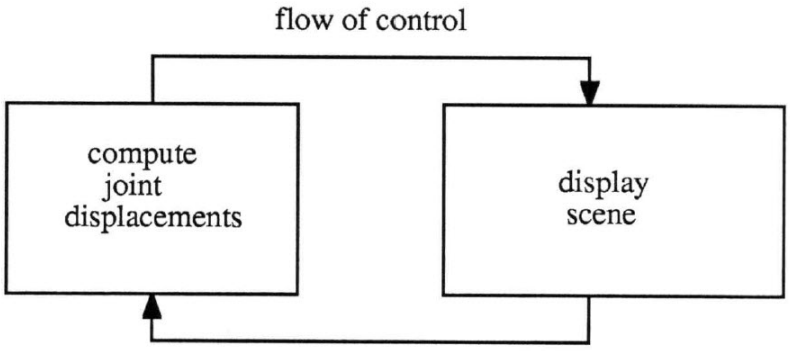


Figure 3. 1: Simple animation loop

The joint angles may be calculated within the same program that is generating the animation. The structure for this type of animation is a simple loop. The performance is limited because each block must finish before the other one can start. This structure also limits the modularity of the program because the animation is bound to the kinematics.

A shared memory structure may be used to communicate joint angles to the animation program. This method is only available with operating systems that support multitasking or multiprocessing. A shared memory is a memory segment that has been mapped into more than one process. For the animation of the robotic system the memory segment is shared by the animation program and the inverse kinematics and decision making routines that are generating the joint displacements. This arrangement may be represented as two loops.

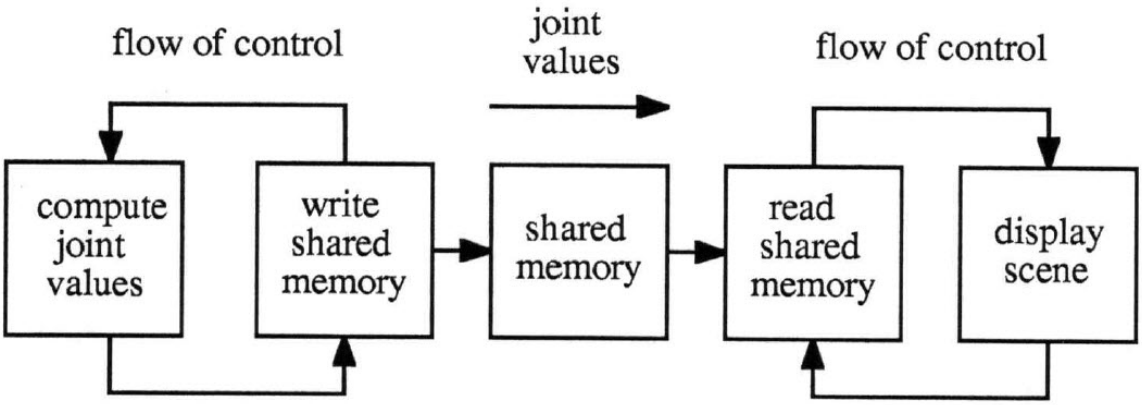


Figure 3. 2: Animation from shared memory

The shared memory allows the inverse kinematics and decision making algorithms to be uncoupled from the animation. Changes can be made to one without having to recompile or relink the other. Allocation of the computer's resources can be left up to the operating system or some priority based scheduling hierarchy might be developed to optimize sharing of the computer's resources. It is important to be sure that both processes do not access the shared resources simultaneously. This conflict may be automatically resolved by the operating system or a semaphore or some other technique may be used. The semaphore gets its name from the railroads where a semaphore is used to prevent two trains from colliding on a shared length of track.

Multiple computers may also be used in the animation. A possible scenario would be for one computer to communicate with a manual controller and generate joint angles while another computer concentrates on generating the graphical images.

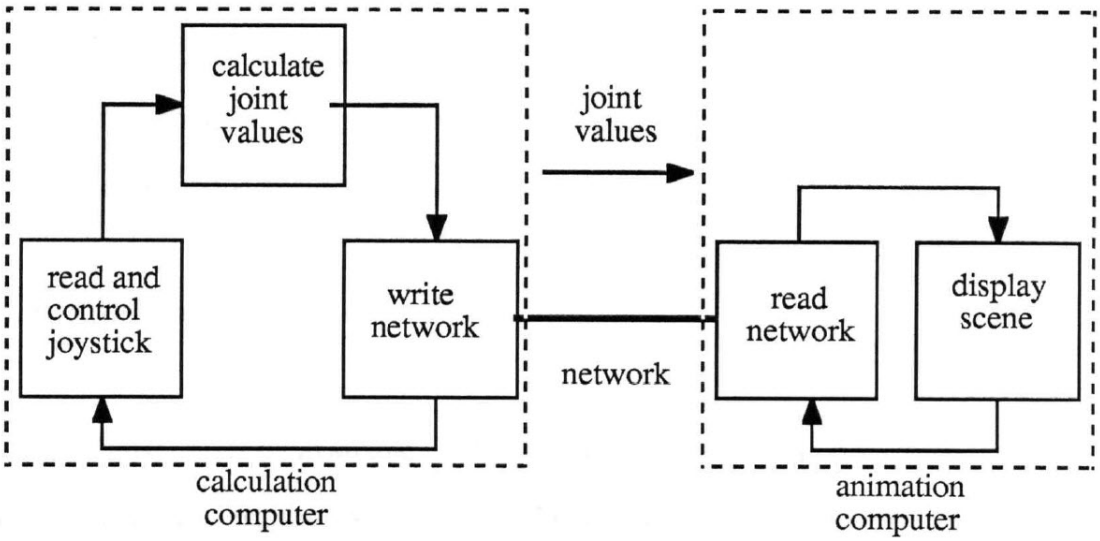


Figure 3.3: Animation with distributed computer processing

The network that connects the computers must be considered. A twenty degree of freedom robot whose joint angles are represented as thirty-two bit floating point numbers requires at least six hundred and forty bits of data to specify the kinematic state of the robot for a single frame. An animation loop rate of thirty frames per second requires a network bit rate of at least nineteen thousand two hundred bits per second.

$$20 \frac{\text{dof}}{\text{frame}} * 32 \frac{\text{bits}}{\text{dof}} * 30 \frac{\text{frame}}{\text{second}} = 19200 \frac{\text{bits}}{\text{second}}$$

This is higher than the bit rate for current RS-232 serial ports, but easily attainable by many other network protocols.

The animation may also be created by cycling the display through a series of preprogrammed positions. The data that specifies these positions may

be stored in data files. The data files may be written in a standard format by inverse kinematics and decision making algorithms, dynamic simulations or by any other robotics research applications where an animated display may be of value, but real-time performance is not necessary.

3.3 Surface Math

Most currently available digital graphics computers use polygonal representation to display solid surfaces. Polygonal representation describes solid surfaces as filled planar areas bounded by points at the vertices. The minimum number of points that can describe a solid surface is three, a triangle. Lighting models often need to know the direction of the surface normal as well as the position of each vertex. Simple geometric relations and computer algorithms can be used to generate the vertex points and the surface normals. Actual curved surfaces, such as a pipe, tend to have a continuous curvature. Polygonal representation can only approximate these continuous curves as many flat surfaces. The more flat surfaces that are used, the closer the approximation comes to a continuous curve. Eventually, the resolution of the display terminal is surpassed and there becomes no visible difference between a continuous curve and a polygonal approximation. An algorithm for defining an open tube gives an example of a polygonal approximation of a continuous curve. This example shows the tube as being comprised of many rectangles. The definition of each rectangle requires four points each with x, y and z coordinates.

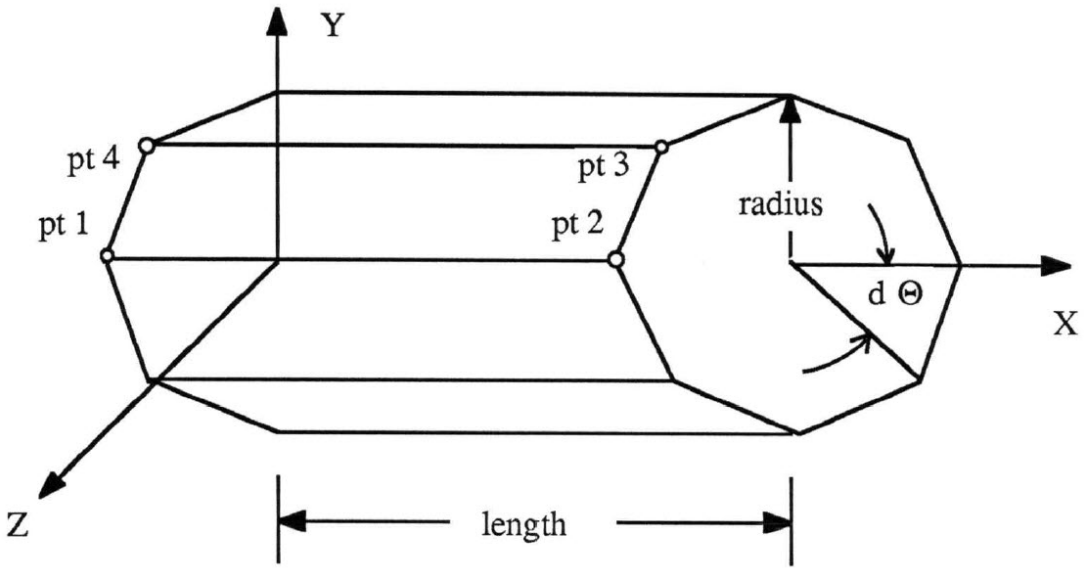


Figure 3.4: Polygonal approximation of a tube

```
for(theta = 0 to theta = TWOPI)
{       point1x = 0. ;
        point1y = radius * cos(theta);
        point1z = radius * sin(theta);
        point2x = length;
        point2y = radius * cos(theta);
        point2z = radius * sin(theta);
        point3x = length;
        point3y = radius * cos(theta + dtheta);
        point3z = radius * sin(theta + dtheta);
        point4x = 0. ;
        point4y = radius * cos(theta + dtheta);
        point4z = radius * sin(theta + dtheta);
```

```
        theta = theta + dtheta;        }
```

Similar software functions can be written to generate many three-dimensional shapes. These shapes include boxes, wedges, tubes, cones, spheres and toroids. These functions can be written to have simple scaling arguments such as diameter, length, height and others in order to create a three-dimensional solid surface graphics library. Translational and rotational transformations can then be used to build more complex objects from the simple functions in the library.

Surfaces that are not easily described by simple primitives can be approximated with triangular meshes. Sets of points on the surface are identified and triangles are then drawn between the points. Triangles offer the finest resolution because three points is the minimum to describe a surface. There is also a computational benefit associated with using triangular meshes. This is because computer graphics typically involves many transformations and operations on each point that is to be displayed. Since two points on each triangle in the mesh also specify two points on an adjacent triangle, the computer only has to transform one new point for each new triangle.

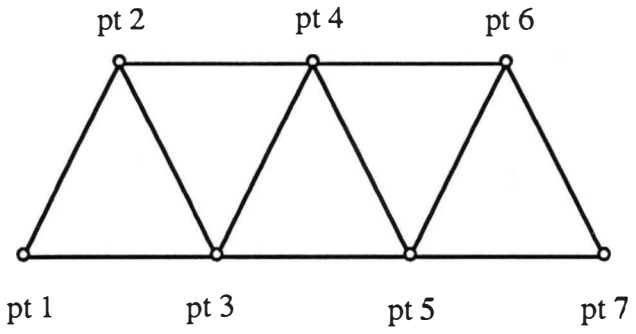


Figure 3. 4: Triangular mesh

As an example, consider a mesh of five triangles. In order to draw the first triangle, 123, points one two and three must be transformed. However, in order to draw triangle 234, only point four now needs to be transformed since points two and three have already been transformed. This type of pattern continues through the mesh. The benefits in computational performance result from the fact that fewer points need to be transformed.

Splines can also be used to approximate curved surfaces. Splines typically maintain continuity of position, slope and the rate of change of slope to approximate a continuous curve.

A lighting model enhances the realism of solid surface computer graphics. A lighting model may use the color, location and orientation of the light source; the surface properties, location and orientation of the surface and the position and orientation of the viewer in order to determine the shading of each point on the surface. Lighting models can be extremely complex. Typical lighting in the real world may come from infinite light sources, such as the sun where light rays may be considered to be parallel, from local lights that diverge and from ambient light that has been reflected and scattered by many other surfaces. The best performance is typically obtained with a single infinite light source. Since the light from an infinite source is parallel, the lighting vector remains constant throughout the scene. Lighting models may also incorporate the surface normal at each vertex. Polygonal graphics that specify surface normals that are actually normal to the surface tend to appear faceted with

distinct color changes between polygon faces. The surface normal may also be specified as an average between the two polygons to give smoother shading.

Generating the polygonal representation typically involves computer algorithms that sweep through some type of geometric relation. Performing these calculations for each display frame represents a computational load. Significant performance benefits can be obtained if these calculations are performed only once and the results stored in fast access memory. The animation algorithms then read the points and normals out of memory without having to recalculate them each time. This results in faster graphics, but the computer code becomes more complicated because a fast access database must be built to include the surface description of all objects in the scene.

3.4 Feature Based Model

There may be other features associated with the model in addition to the information necessary to produce the actual visible display. The range of motion possible for each joint should be accounted for in an animated simulation. The forward kinematics is a feature that is associated with the animated display. Dynamic features may also be associated with the model.

Joint limits are an excellent example of the need to specify the robot model as completely and accurately as possible. If the joint limits are not accounted for in the model, then it may be possible to have the animation appear to be fine while the actual robot has reached a physical joint limit. Attempting to drive the joint past the limit could damage the robot. Joint limits may be

incorporated into the animation model at the modular level. Visible warnings can be given when the joint limits are reached.

Forward kinematics is another feature that may be incorporated into the animation of robotic systems. By incorporating the forward kinematics as a feature of the model, the state of the robot for each frame is determined by the value of the joint variables. This is similar to an actual robot where, neglecting compliance, the visible state of the robot is determined by the angles at the revolutes and the lengths of the sliders. The forward kinematics for the animation is determined by the topology and content of the system, in other words, which types of joints are present and how they are connected. The topology remains constant while the values of the joint variables is changed to produce the animation effect.

It is also possible to associate dynamic features with the animation. The addition of dynamic features would result in a much more complex model, because the topology of the system and the values of the joint variables no longer completely specify the state of the robot. Deformations would cause the appearance of the display to change; for instance links might bend and oscillations could develop. Information about the dynamic state of the robot might be conveyed to the viewer through surface shading and coloring. A dynamic model that includes compliances, damping and mass content could be used to create an animation that receives joint torques and external loads as input rather than simply joint displacements. The correctness of the animation would then be dependent upon the accuracy of the dynamic model.

3.5 Precision

In many respects the computer animation of robotic systems is an instrumentation application. There is a transduction process where numerical information is translated into visual information that is used to evaluate the system state and performance. As with most instrumentation problems, the question of precision is of extreme importance. The precision, as well as the resolution and range of the animation, should be examined. The animation instrument has two distinct parts: the numerical representation and algorithms that drive the display and the display terminal itself.

The precision of an instrument is the number of distinguishable different states the output may assume. For the case of an object on a computer screen, a reference point on the surface of the object might occupy any of the pixels on the screen. Current graphics computers have on the order of one thousand by one thousand, or one million different pixels that the reference point may occupy. The precision of the computer algorithms depends upon the internal numerical representation. In the case of floating point numbers that are represented by one sign bit, seven bits of exponent and a twenty-four bit mantissa, the precision is twenty-four bits or seven decimal digits. Round-off and truncation error can also affect the numerical precision, however if the numerical precision is a problem, higher precision numerical representation can be used. Thus it is seen that the limiting factor of the precision of the animation is the display terminal.

The range of the instrument is the difference between the minimum output state and the maximum output state. The range of a terminal that has one thousand pixels on each axis is simply one thousand pixels in the horizontal and vertical directions. The range of the numerical representation again depends upon the specific type of numbers that are used for the computation. Citing the example of a floating point number with one sign bit, seven exponent bits and a twenty-four bit mantissa, the range is from $2.\ 7 * 10^{-20}$ to $9.\ 2 * 10^{28}$ and zero.

The resolution of an instrument is the smallest change in input that will result in a predictable change in the output. The output of the computer animation of a robot is a visual image of the robot, and the input is the value of all of the joint variables. The resolution of the animation is the smallest change in the value of a joint variable that can be reliably seen on the display. Complicating the resolution of the display is the possibility of mapping different scales of dimensions onto the computer screen. Assuming, for instance, that a robotic workcell that is three meters on each edge is mapped onto a display terminal that is one thousand pixels on each axis, then the resolution is three millimeters. The resolution could be increased by mapping a smaller portion of the workcell onto the computer screen, however this would decrease the range of view. The resolution of the numerical representation of the robot within the computer is highly dependent upon the algorithms that are used to generate the surface description. When using polygonal representation, the number of flat sides used to approximate a curved surface is an important consideration. Consider a tube of radius r with n flat sides. The radius covers some number of

pixels, p. The distance between each pixel is therefore r/p . It is possible to calculate the number of flat sides that the object must have so that the polygonal representation is not within the resolution of the screen. The difference between the actual radius and the polygonal approximation of the radius must be less than the resolution, r/p.

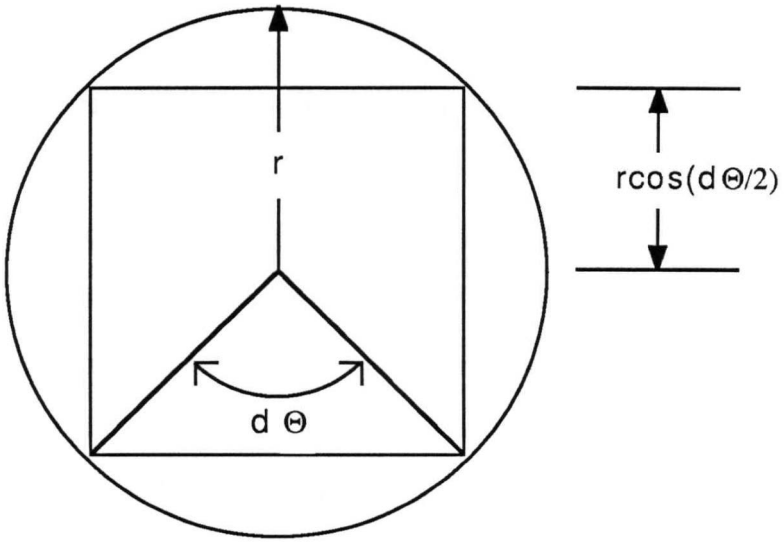


Figure 3. 6: Polygonal approximation of a circle

$$r - r\cos\left(\frac{d\theta}{2}\right) < \frac{r}{p}$$

$$1 - \cos\left(\frac{d\theta}{2}\right) < \frac{1}{p}$$

$$1 - \cos\left(\frac{pi}{n}\right) < \frac{1}{p}$$

$$\arccos\left(1 - \frac{1}{p}\right) < \frac{p\imath}{n}$$

$$n > \frac{p\imath}{\arccos\left(1 - \frac{1}{p}\right)}$$

Chapter 4
Specific Developments In This Report

The incorporation of modularity and reconfigurablility into the computer animation of robotic systems facilitates the application of computer animation to many areas of robotics research. Animations of an extremely large class of robotics systems can be created by simply assembling together modules from a finite set of one, two and three degree of freedom joint modules and generic links. The hierarchical nature of robotic systems makes the interpretation of kinematic data in numerical form very difficult. Computer animation is an effective method of visually presenting kinematic data, such as might be generated by obstacle avoidance algorithms, redundant inverse kinematics routines and as the output from dynamic simulations. If the computer is continuously animating data as it occurs in response to external events, the animation may be said to be running in real-time. The ability to run in real-time allows the animation to be used in the development of manual controllers. Modular and reconfigurable animation can be used in the development of serial, parallel, mobile and hybrid manipulators and is ideally suited to the

development of modular and reconfigurable robotic systems. The ability of the computer to mathematically scale the display to a very fine resolution allows the animation to display precision operation. A modular approach to animation may be used to generate world model databases that include multiple robots and obstacles that may be fixed to the universe frame or may move about in the environment to simulate task performance.

4.1 Obstacle Avoidance

Obstacle avoidance refers to moving the robot about in the environment while avoiding collisions with objects in the environment. The obstacles may include the robot itself, fixed objects in the environment, moving objects in the environment and also other robots that may be operating in the same workspace [43]. The cost of an industrial robot colliding with an obstacle in the environment could be very high, both in terms of repair to the robot and the environment as well as the cost of lost productivity due to downtime. The results of a collision in a more sensitive environment, such as space, nuclear or military applications could be enormous. The cost of such collisions makes obstacle avoidance of fundamental importance when planning the robot's path.

The schemes for performing obstacle avoidance might loosely be divided into two groups, those that address obstacle avoidance from a local point of view and those that address obstacle avoidance from a global point of view. Global methods evaluate the robot and the environment and attempt to find a continuous collision-free path from a starting position to a goal position.

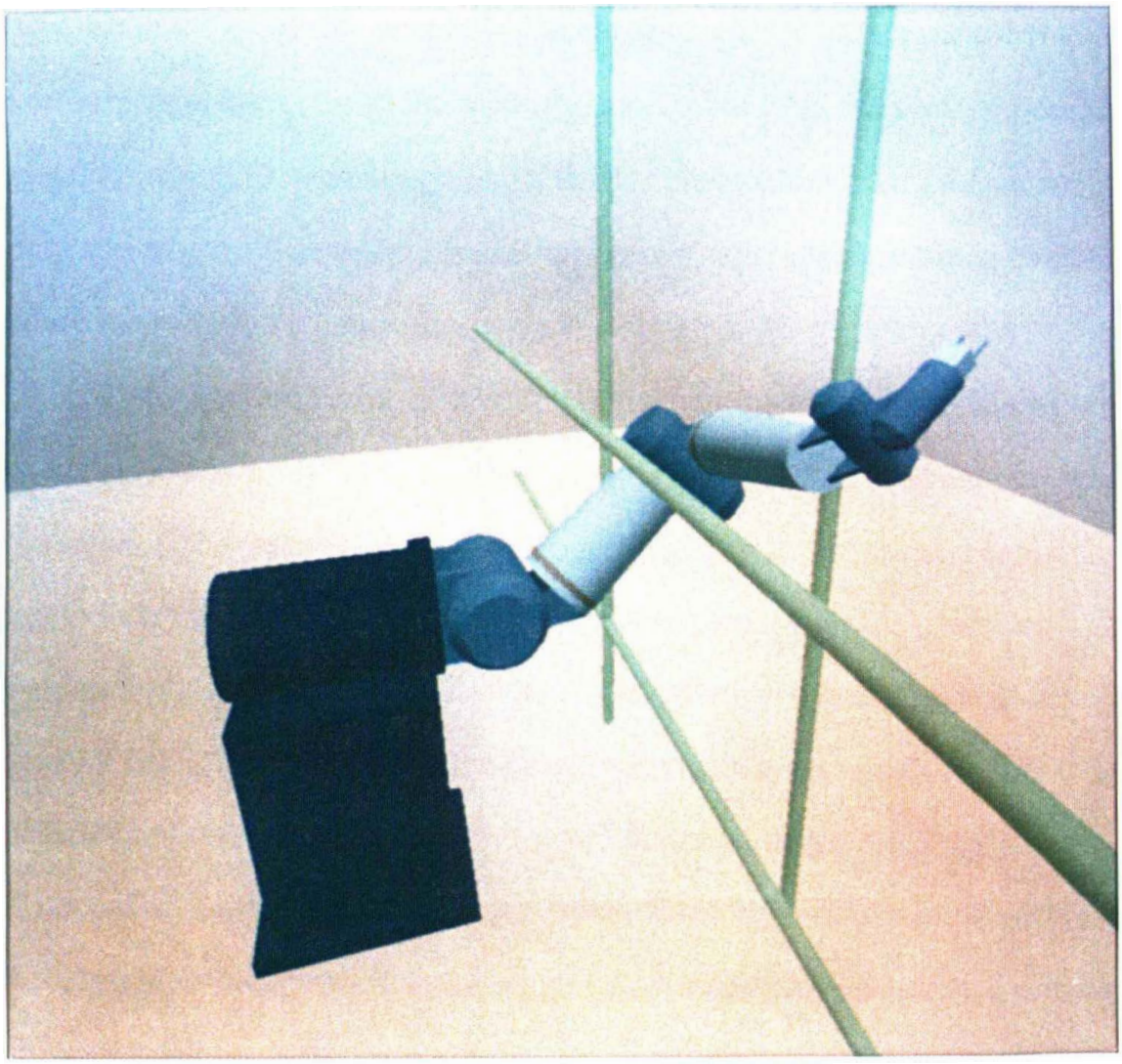


Figure 4.1: Robot performing an obstacle avoidance maneuver.

A common approach to global obstacle avoidance is to arrange the obstacles as vertices in a graph [14]. Collision-free paths to the goal are found as a sequence of edges along the graph. Local approaches examine the robot and the environment at each step as the robot follows a path from the starting position to the goal position. Imaginary potential fields may be created around the robot and the obstacles. The positions and overlaps of these fields are then evaluated at each step along the path.

Computer animation of modular and reconfigurable robotic systems can be used as a tool in the research of both global and local obstacle avoidance techniques. The same database that is used to generate the modular robot and the modular environment can also be used as a database for the obstacle avoidance algorithms. The animation can also show the boundaries of any potential fields that are used by the obstacle avoidance routines. Moving obstacles can be created by attaching to the obstacles degrees of freedom referenced to the universe frame. A robot performing a task, such as pick and place, might be simulated in this manner. A valve might be simulated as a two degree of freedom screw joint attached to the environment. The robot opening or closing this valve could be simulated by specifying the proper joint displacements.

4.2 Redundant Robots

A redundant robot has more degrees of freedom than are necessary to specify the state of the end effector. Any robot with seven or more degrees of

freedom is redundant because it only takes six parameters to specify the position and orientation of an object in three-dimensional space. The extra degrees of freedom can be used to improve the performance of the system by allowing the end effector to reach the goal with many different joint paths. Different criteria can be used to determine which joint paths are best for a given situation [5]. The inverse kinematics are complicated by the fact that the Jacobian of a redundant robot cannot in general be uniquely inverted. Other methods must be used to determine what to do with the extra degrees of freedom. Walking machines, multifingered grippers and "snake" manipulators are all examples of redundant robotic systems.

Many different methods of incorporating the extra degrees of freedom into the inverse kinematics algorithms for redundant robots have been developed. Since there are many different joint positions that will result in the same end effector position, redundant inverse kinematics is often approached as an optimization problem where many different criteria are considered. Local methods solve the inverse kinematics at each discreet point along the path, while global methods consider the path as a whole. A pseudoinverse of the Jacobian matrix is often used to solve for the inverse kinematics of the redundant robot [30]. The pseudoinverse does not in general give a unique solution. Thus, cyclically following the same end effector path does not guarantee repeating the same joint paths. Multiple criteria have also been incorporated to resolve the kinematic redundancy. These criteria include singularity avoidance, obstacle avoidance, robot dexterity, energy minimization, manipulator precision, load carrying capacity, speed of operation and others [5]. Complicating the

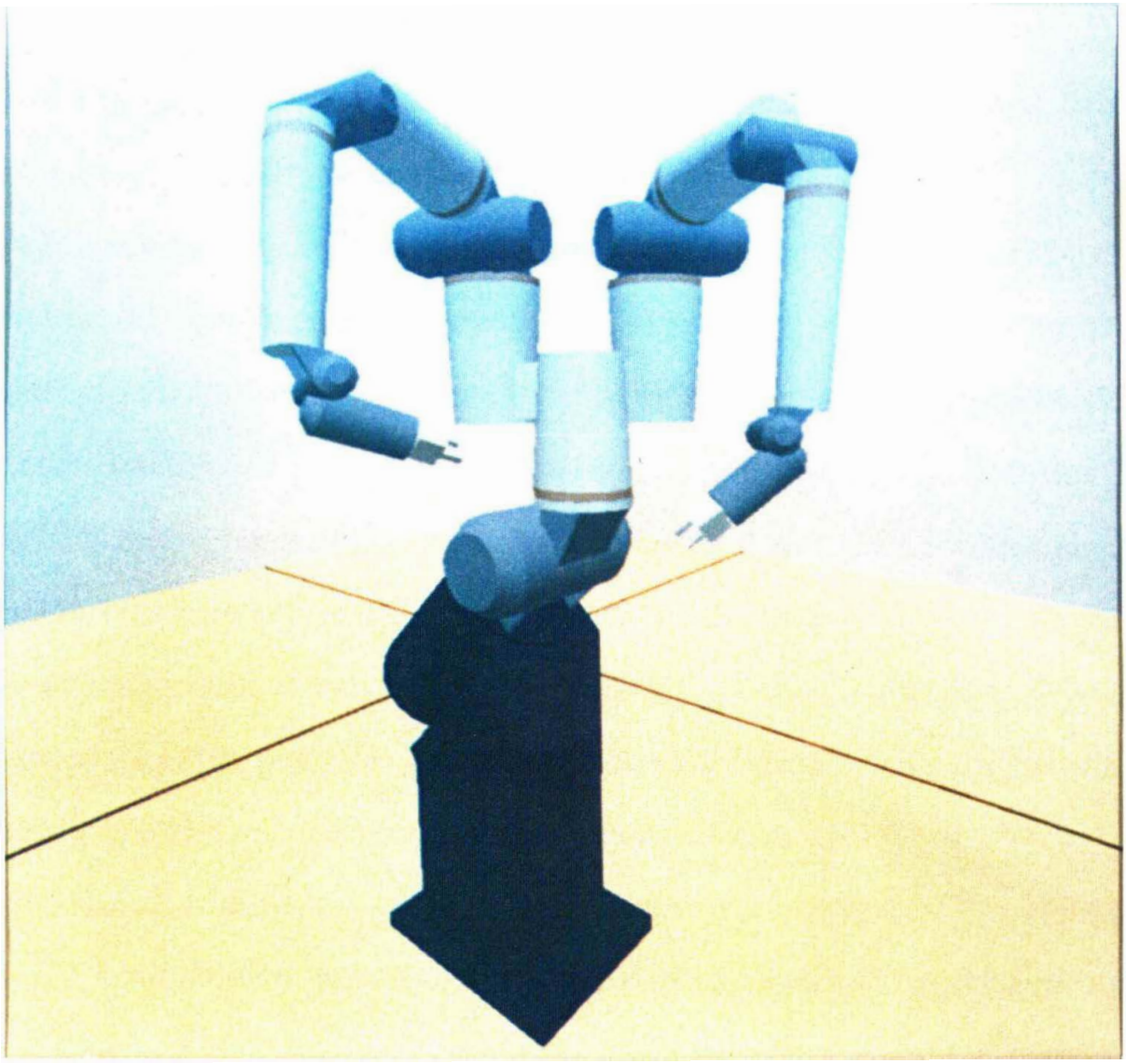


Figure 4.2: Seventeen degree of freedom redundant robot

incorporation of multiple criteria is the fact that the multiple criteria can be in multiple energy domains, have different units and be of very different magnitudes. This requires that the criteria be properly normalized and prioritized [5]. Another method of solving redundant inverse kinematics breaks the redundant chains into smaller chains that are not redundant and may be solved by direct inversion of the Jacobian [18]. The solution is constrained such that the cut ends have the same spatial position, velocity, and acceleration. The position and trajectory of the cut point may be chosen such that the resulting inverse kinematics solutions are well behaved, or by other decision making or intelligence schemes. Genetic algorithms have been used to find inverse kinematics solutions for redundant robots [34]. Genetic algorithms are search procedures that are based on genetics and natural selection. A simple genetic algorithm generates solutions and then chooses which algorithms will propagate based on their fitness, where fitness is the maximization of some performance criteria.

By definition, redundant robots have extra degrees of freedom. More computer programming time is necessary to generate the forward kinematics and visual simulation for these extra degrees of freedom. This time is significantly reduced by using computer animation based on a generalized modular and reconfigurable mechanical architecture. A robot with extra degrees of freedom simply has more joint modules and generic links. The extra degrees of freedom makes the interpretation of kinematic data in numerical form even more difficult, thus adding to the benefit of displaying this data graphically.

The many criteria that may be used by the inverse kinematics schemes may also be presented graphically by means of computer animation.

4.3 Dynamic Simulation

Dynamic modelling of robotic systems is an active area of robotics research [4] [6] [21]. Dynamic modelling incorporates mass, compliance and damping to simulate the system response to joint forces and external loads. Computer animation can be used to display the results from these dynamic simulations.

Dynamic models can be used to evaluate the dynamic condition and performance of a robot as it performs a task. The condition number of the Jacobian matrix has been suggested as a performance measure [38]. Singular values of the Jacobian may also be incorporated into an evaluation procedure [53]. The state of the inertia matrix has been used as a measure of the robot's dynamic ability [23]. The dynamic model can be incorporated into feed forward control algorithms. Feedforward control incorporates the system dynamics into the control algorithm in order to synthesize the output. The dynamic model for the robotic system may be developed in terms of kinematic influence coefficients [48]. The kinematic influence coefficients employ the geometry of the robot to relate the system's dynamic characteristics as they appear at any input to the system.

Computer animation is by nature kinematic. An animated robot can exhibit high speed, high precision, no deformation and no backlash. This is,

Research Application	Desirable Attributes of Animation
Obstacle Avoidance	available database, solid surface display, multiple viewing angles hidden surface removal
Redundant Robots	modular creation of animation, criteria display
Dynamic Simulation	incorporate dynamic parameters, display of performance measures
Real-time Control	animate continuously as data is produced
Manual Controller	animate continuously as data is produced, ghosting capability, criteria display, multiple viewing angles
Control in the Small	solid surface display, multiple viewing angles, ability to "zoom-in"
World Model Database	available database, multiple viewing angles, hidden surface removal

Table 4.1: Desirable attributes of an animation package for various research applications.

of course, a tremendous simplification. By coupling a dynamic model to the animation it is possible to make the robot appear to move in a more realistic manner. A modular and reconfigurable robotic architecture presents an excellent opportunity to combine dynamic simulation and computer animation. Constructing the graphical robot on the screen defines the modules that make up the robot and the geometry . The dynamic description can be stored as a feature within the modules. As each module is added to the robot, the dynamic equations could be automatically generated using the geometry and the dynamic description of the modules. The incorporation of dynamic analysis with real-time control can be studied with computer animation by running the animation continuously on the data as it is being generated by the control algorithms. The animation also provides a method of graphically displaying performance measures as the robot performs a task.

4.4 Real-Time Control

The real-time control of robots has been the subject of many recent robotics research papers. A good definition for real-time control in robotics seems to be "while the robot is moving". In other words, the real-time controller continuously sends commands to the robot and operates on sensory information as the robot performs its task. By this definition the difference between recent real-time control schemes and previous controllers is that the more recent real-time controllers are based on digital computers and

microprocessors.

Recent work in real-time control has included the development of real-time computer architectures and real-time programming languages [27] [36] [40]. These real-time control schemes dynamically allocate the computers resources in response to external events. The events may be generated by external sensors that provide information about the environment and the state of the system, or the events may be generated internally perhaps in response to some calculation reaching completion. The response of the controller to events is typically priority-based [27]. An important consideration with real-time controllers is the context switch time. The context switch time is the amount of time it takes the computer to stop what it is doing, save necessary data and begin responding to another event.

A computer animation that is continuously animating data as it is produced is running in real-time. The data may be produced by another process that is residing on the same computer, or the data may be produced externally by another computer or a manual controller. The data might also be generated by the robot itself with strain gauges, resolvers, etc. The animation can also take desired data from the controller and actual data from the robot and display the difference between the two. By running the animation in real-time the computer can become an animated robot testbed.

4.5 Manual Controller Development

The manual controller is the interface between the man and the machine. This interface should support two-way communication. That is, the human should be able to send commands to the robot and the robot should be able to relay sensory information back to the operator. The interface is complicated by the need to transform between the robot geometry and the controller geometry. There exists the possibility of replacing the manually controlled robot with a fully autonomous robot. There are, however, many benefits to be obtained by keeping the human in the loop. These benefits include the ability of the human to provide redundant control in cases of autonomous control system failure, unexpected changes in the operating environment and tasks too stringent to be addressed by the current state of the art in autonomous control [41].

Modern manual controllers attempt to provide a feeling of telepresence. In addition to visual information there is a tremendous amount of proprioceptive feedback when a human physically performs a task. This feedback includes net forces and the distribution of forces. An articulated manual controller with actuated joints can be used to provide force feedback.

Modern robots incorporate multiple sensors that provide information about the system state and environmental conditions as the robot is performing its task. The computer animation can be used to provide a dynamic graphical display of this information. The animation can be used to provide common

visual information, such as the positions and motions of the robot and objects in the environment . The animation can also provide information about many other system criteria, such as force distribution, stress distribution and energy consumption. The animation can display multiple viewing angles and can warn of impending collisions or close approaches between surfaces. By incorporating a dynamic model the animation can predict how the robot will respond given input commands from the manual controller. These input commands can be modified until acceptable results are obtained and then be given to the actual robot. The modular and reconfigurable architecture can be used to rapidly build many different robot configurations. These animations can then be used to evaluate the performance of universal controllers that are not dependent upon the robot geometry.

The computer can also be used to display animated ghosting of the robot manipulator. Ghosting refers to displaying two images of the manipulator in the same scene. One image could be used to display the actual position of the manipulator while the other image, the ghost, could be used to display the desired position.

4.6 Serial Robots

A serial robot is created by connecting a number of one degree of freedom joints in a series. The output of each joint is connected to the input of the next joint by a link. This sequence continues until the end effector is reached. The end effector terminates the serial robot and is used to interact with the environment in the performance of the robot's task. The serial structure results in a hierarchy where the movement of one joint changes the position and/or orientation of all distal joints and links. The serial structure also propagates errors throughout the system. Errors that occur at an interior location may be amplified by the robot's geometry and show increased effect at the end effector. The serial manipulator must also support the load of its own actuators, or suffer decreased precision due to the compliance inherent in torque transmission devices, such as torque tubes and tendon drives. The serial manipulator tends to have a large workspace and good dexterity.

A representative dynamic model must be used to precisely control the serial manipulator. The model may be calculated using kinematic influence coefficients which are based upon the geometry of the system [48]. The dynamic equations can be formulated in a manner such that they may be solved in real-time [50]. Many current control schemes utilize feedback in an attempt to cancel errors at the output. The incorporation of a dynamic model allows feedforward control which utilizes system dynamics to synthesize the input

function from the desired output function. Feedback can then be used to cancel errors due to an imperfect model.

A robot animation package based on a modular and reconfigurable robotic architecture allows animations of an extremely large class of serial manipulators to be quickly assembled. These animations are created by simply connecting joint and link modules one after another. The number of degrees of freedom that the model may have is ultimately limited by the memory capacity of the computer, but thousands of degrees of freedom are possible with a modular architecture and current computers. "Snake" robots can be created by connecting together many joint modules with very short links. The modular computer animation naturally incorporates the hierarchy which determines the forward kinematics of the serial robot. Many different possible robot configurations can be graphically constructed and the animation evaluated before beginning the construction or assembly of the actual robot.

4.7 Parallel Robots

The joint angles of a parallel robot may not all be specified independently. A four bar planar mechanism, for example, has four joints but only one degree of freedom. By specifying the angle of one joint the angles of the other joints are constrained. The parallel robot typically has a smaller workspace than the serial robot, but there are also many benefits associated with using a parallel structure. The parallel structure can utilize mechanical advantage to increase its load carrying capacity. The parallel robot allows a

choice among the joints as to which ones will be used as the inputs to the system. Several degrees of freedom can be obtained while still allowing direct actuation with the actuators fixed to the robot base [6]. Redundant actuation is also possible with parallel structures. The redundant actuators can provide fault tolerance and antagonistic actuation. Antagonistic actuation shows promise for high-precision tasks that require disturbance rejection.

Kinematic influence coefficients can be used as a general method of the kinematic and dynamic analysis of parallel structures [4]. Kinematic influence coefficients allow the dynamic and kinematic description of the system to be referenced to any joint within the parallel chain that might be used as an input to the system. This is of particular benefit for applications that involve redundant actuation for fault tolerance or antagonistic control. Computer programs have been developed that automatically generate kinematic solutions and dynamic simulations of parallel structures [21].

Computer animation of modular and reconfigurable robotic systems can be used to animate parallel structures. The parallel structure is simply built from open chains. The chain is closed to form a parallel structure by properly specifying each joint angle. This method for creating the animation is of particular benefit for the animation of data that has been generated by kinematic and dynamic analysis that treat the parallel chain as a set of constrained serial chains [4]. The parallel chain is kinematically cut, and the resulting serial chains are constrained to have zero relative position, velocity and acceleration at the cut. Errors in the kinematic solution are illuminated during the animation because the chain will appear to break at the cut.

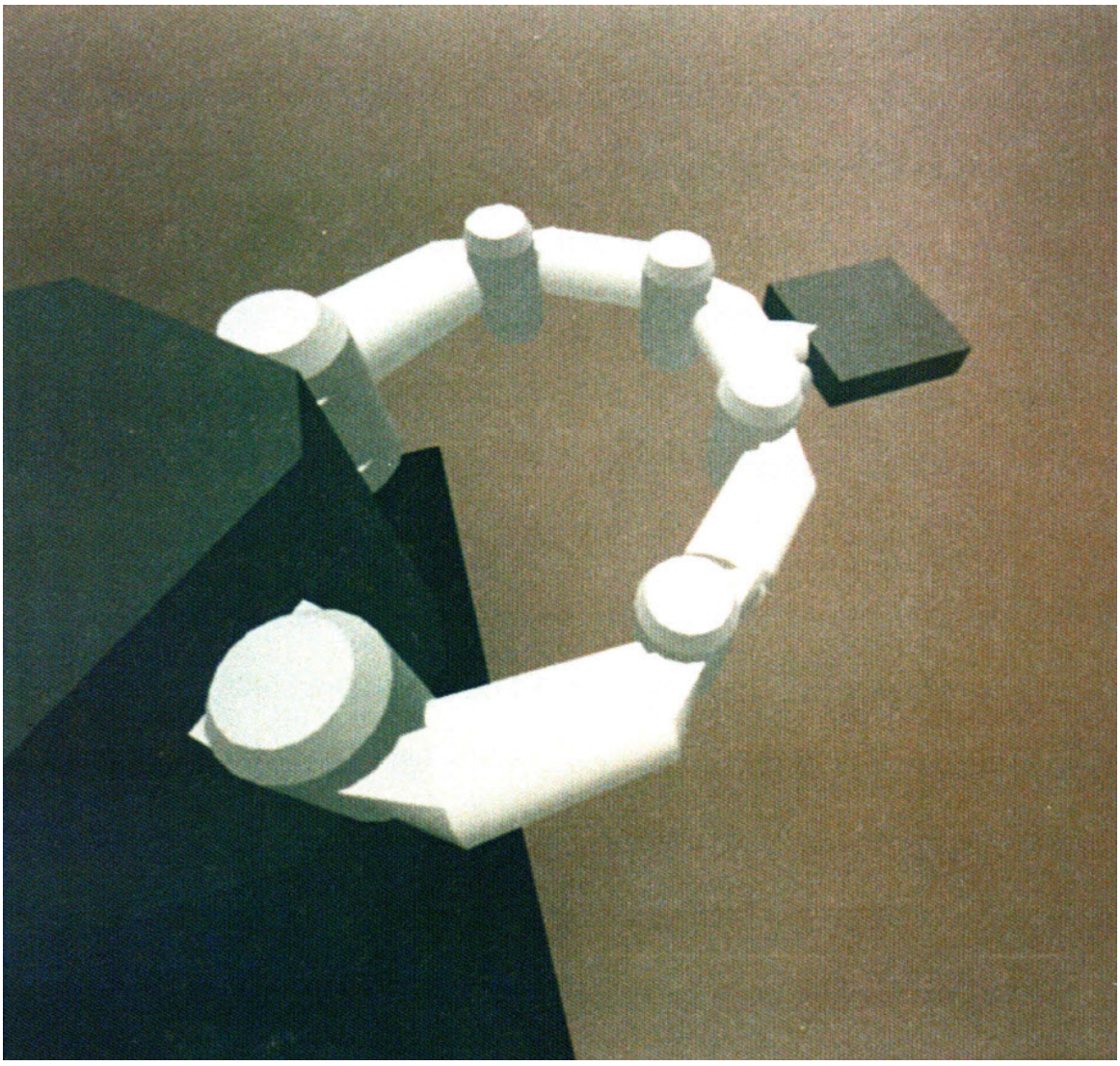


Figure 4.3: A parallel three DOF planar manipulator with six revolute joints

4.8 Mobile Robots

The mobile robot is able to change its base location within the environment. The mobile robot cannot, however, be expected to change its position or orientation in an arbitrary manner. The base frame of a wheeled robot, for instance, may be constrained to move in a plane. The motion of the mobile robot may also be restricted by objects in the environment. In order to plan the path of the mobile robot it is necessary to characterize the environment and the motion constraints of the robot.

The mobile robot must somehow characterize the environment in which it is to operate. One method would be to preprogram a description of the environment and store it in the robot's memory. If the environment is not completely known or is changing, then the robot must use sensors to obtain information about the environment. The sensors may include cameras, sonar, lasers and radar. The sensor data may then be used to create a map of the environment. It is also possible to use a preprogrammed database to define the objects as being in the environment and also to use sensors to identify and locate the objects . Information about the environment can be stored in a tree representation referenced to a known location. The obstacles and objects in the environment can then be referenced to the known location [8]. Various schemes have been developed to search the tree for a collision free path.

Computer animation based on a modular and reconfigurable mechanical architecture may be readily applied to the animation of mobile robots. The

mobile robot is created by simply attaching the appropriate degrees of freedom to the robot base and referencing them to the universe frame. Specifying the value of the joint variables then locates the robot's base frame within the environment. Objects and obstacles in the environment can be created with the joint and link modules. A changing environment can be created by attaching degrees of freedom to the modules and referencing them to appropriate points in the environment. The viewing parameters of the animation can be varied to follow the animated mobile robot as it moves about in the environment.

4.9 Hybrid Robots

A hybrid robotic system contains both parallel and serial structures in an attempt to gain the benefits associated with each. The parallel components contribute strength and rigidity while the serial components add dexterity and reach. In excess of thirty percent of the robots in use today incorporate at least one parallel structure [12]. The Cincinnati Milacron t3776, for example, incorporates an inverted slider crank at the shoulder and wrist.

The algorithms to control the hybrid robot must be sufficiently general to handle both the serial and parallel structures. The parallel components add complication because the joint variables may not be controlled independently. The use of kinematic influence coefficients allows the kinematic and dynamic description of the system to be referenced to whichever joints are specified as the inputs.

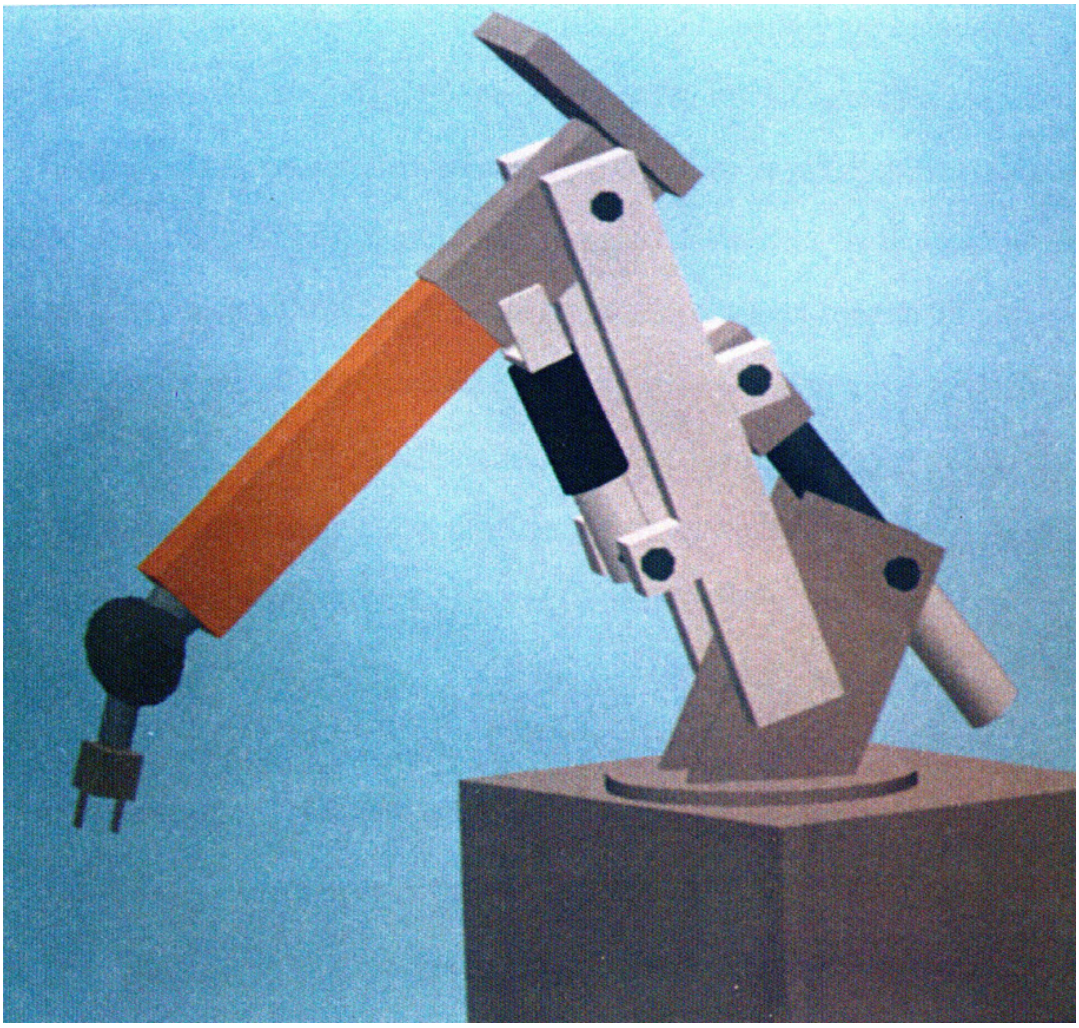


Figure 4.4: A hybrid robot that contains two in-parallel chains

Hybrid systems can be graphically created and animated using the generalized modular mechanical architecture. The modular animation is created by simply specifying which modules are used and how they are connected together. More than one chain can be connected to a link to create a branching effect. The extra chain can then be closed to create an in-parallel structure or left open to animate multi-arm sytems. Certain modules within the architecture, such as the three degree of freedom spherical shoulder, are inherently parallel. Hybrid systems will also be created by simply incorporating these modules within a serial chain. The equations which define the interior joint angles are stored as a feature of the parallel modules. This way only the value of each degree of freedom must be specified and the interior joint angles can be calculated automatically.

4.10 Control in the Small

Control in the small refers to small high-resolution position and force control superimposed upon the larger motions of the robot about its workspace. Control in the small can be used to compensate for errors in the robot's motion due to friction, backlash, compliance and inertia. A serious consideration in the use of control in the small to provide precise compensation for errors is the need to precisely characterize the errors themselves.

The addition of a micromanipulator between the terminal link and the end effector of the robot has been proposed as a means of providing control in the small [2]. The resulting system possesses both the large scale motion

capabilities of the standard robot and the precise control characteristics of the micromanipulator. Control schemes have been developed for this type of robotic system [20]. These control schemes are reliant upon precise endpoint sensing. This may be a reasonable proposition for operation in a plane, however, precise spatial position and orientation tracking of a robot's endpoint is currently a very expensive and difficult task.

Computer animation is well suited to the visual simulation of the motions of robots possessing control in the small capabilities. Simple scaling operations can be used to mathematically magnify the display to a very fine resolution. Multiple views of the scene may also be animated simultaneously. One view could be of the entire robot from a fixed point in the environment while another view could follow the end effector motion with a very fine resolution. Both the macroscopic and microscopic characteristics of the robot with control in the small could then be displayed simultaneously.

4.11 World Model Databases

A world model database is information about the environment that is stored in the robot's memory. The robot can access this information while interacting with its environment and also use it for path planning with obstacle avoidance. The model must be complete enough to allow the robot to successfully perform its task, yet unnecessary detail will slow the interpretation of the model.

The world model may be described in many different ways. One possible approach involves dividing the world into a three dimensional grid.

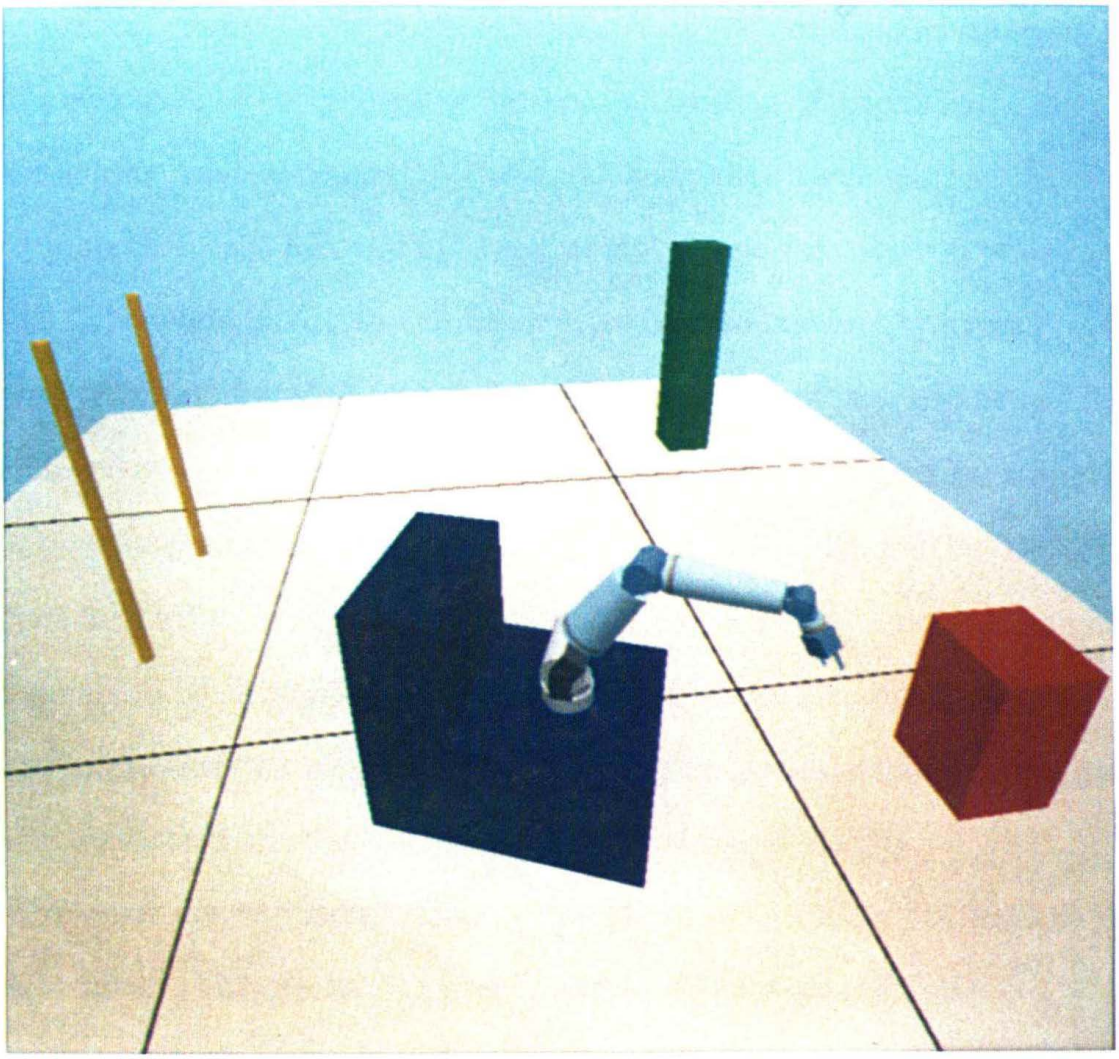


Figure 4.5: A mobile robot in a simple world model

Unfortunately a ten meter cubical workspace described in this way with a resolution of one millimeter has 10^{12} elements. This is quite a large data base to be repeatedly faced. Other methods of describing the world model are more object oriented. A list or graph of the type and location of objects in the environment can be created [8]. The world model may be entirely preprogrammed and assumed to be unchanging during the operation of the robot. The world model may also be created from sensor data by a robot performing terrain mapping operations. The two methods may also be combined by using sensor data to locate known objects within the environment.

Modular computer animation may be applied to either grid based or object oriented types of world model description. The grid-based approach may be simulated by creating a module that is the size of one element in the grid. The model is then visually displayed by putting a copy of the module in all grid locations that are described as being filled. The object-based environment may be displayed by creating modules that match the objects in the environment. The objects then may be positioned and oriented in the environment to create static obstacles, or moving objects may be simulated by attaching appropriate degrees of freedom to the obstacles and referencing them to points in the environment.

Chapter 5
Conclusions

5.1 Summary of Report

This thesis has examined the computer animation of modular and reconfigurable robotic systems. A finite set of one, two and three degree of freedom joint modules and generic links forms the basis of a general mechanical architecture. These modules may be scaled and assembled to form an extremely large class of robotic systems. Computer animation is currently used in the programming and simulation of existing industrial robots. There are, however, many benefits to be gained by the incorporation of modularity, both in the computer animation of the robotic system and in the design of the actual robot. Computer graphics techniques for the animation of solid surfaced objects have been considered. Finally, the application of modular computer animation to many current robotics research topics has been discussed.

environment, or workcell. Process machines, tools, parts and any other objects that the robot will interact with are also placed in the workcell. Computer animation can then be used to simulate the motions of the robot as it performs its task. Off-line programming uses computer animation as a substitute for the actual robot while developing motion control programs. This allows the actual robot to remain in service while the programs are being developed and thus decreases downtime. Research program promotion is another current application of computer animation. Computer animation can be used to convey ideas and concepts while also enhancing a company's or research program's high-tech image.

Currently available graphics workstations can produce animations of a robot moving in a complex environment. These images may be displayed in three-dimensional shaded color with hidden surfaces removed. Unfortunately, writing the computer programs for animations is complex and time-consuming, often requiring thousands of lines of code to produce a single animation. Solid modelling programs are available to ease the creation of computer animations. These packages typically employ solid modelling primitives such as box, wedge, tube, cone and sphere to construct the graphical robot. This is easier than writing computer code, but still takes a significant amount of time as well as requiring experience with the specific solid modelling package that is being used. The incorporation of a generalized modular mechanical architecture greatly simplifies the creation of computer animations of robotic systems. The modular computer animation may be quickly constructed with objects that the robotics engineer is familiar with, such as links and joints. A modular approach

to computer animation allows increasingly complex systems to be quickly constructed and easily used by the design engineers involved in the development of new robotic systems and technologies.

A generalized modular mechanical architecture has much to offer the field of robotics. Modular robots may be reconfigured to perform many different tasks, thereby reducing the threat of obsolescence and reducing costs. The design of modular robotic systems is simplified because the design parameters may be broken down and addressed in small groups that are contained within each module. A modular architecture facilitates the integration of new technologies by allowing the technology to be incorporated into the design of a single module without requiring that the entire system be redesigned.

Computer graphics provides a means of visually simulating a robotic system. If many robot images are displayed in rapid succession with the joint displacements changed by a small amount each time, the robot image will appear to be moving in a continuous fashion. The joint displacements can be generated from within the same program that is generating the graphical display or they may be generated by another process and then passed into the graphics program through a shared memory structure. Higher performance can be obtained by generating the joint displacements on a separate computer and then passing them to the graphics computer over a network. For applications that do not require real-time operation, the joint displacements may be calculated and stored in a datafile for animation at a later time. As the animation will be used to

evaluate the system's state and performance, it is important to understand the precision, resolution and range of the display.

Producing an animated robot display requires a surface description of the robot. Currently available graphics computers use polygonal representation to approximate curved surfaces. Three-dimensional transformation and projection operations are used to locate the surfaces in the displayed scene. The generalized mechanical architecture allows many features to be associated with each module. These features may include both kinematic and dynamic properties.

The incorporation of modularity facilitates the application of computer animation to many current areas of robotics research. The animation may be used to visually present kinematic data such as might be generated by obstacle avoidance algorithms, redundant inverse kinematics routines and from dynamic simulations. The ability to run in real-time allows the animation to be used in the development of manual controllers. Modular and reconfigurable animation can be used in the development of serial, parallel, mobile and hybrid robotic structures. The ability of the computer to mathematically scale the display to a very fine resolution allows the animation of precision operation. Modular animation may also be used to display world model databases that include multiple robots as well as fixed and moving obstacles.

Further research may include the extension of computer animation of modular and reconfigurable computer animation to include the dynamics of the robot as well as the kinematics. Possible dynamics research areas include the automatic generation and solution of rigid-body dynamic equations, the

extension of the model to include joint and link compliances and the concurrent display of information concerning the dynamic state and performance of the system. Of these, the rigid-body dynamics and the display of dynamic state and performance information seem the most feasible with current levels of computer technology.

5.2 Further Research Opportunities and Recommendations

A software package for the computer animation of modular and reconfigurable robotics systems has been developed in conjunction with this thesis. The program employs a graphical user interface to select from among one, two and three degree of freedom joint modules and generic links. These modules may be manipulated on the screen to create a three-dimensional color computer animation with solid surfaces and perspective or orthographic viewing. The interactive, menu-based and mouse-driven program is able to generate computer animations of serial, parallel, mobile and hybrid robotic systems. The animation effect is achieved by accepting data from the keyboard, from datafiles or from a shared memory structure.

The program generates a feature based model and is currently able to automatically perform the forward kinematics of the system. At its current level of development, the computer animation of modular and reconfigurable robotic systems may be applied to the many areas of robotics research that have been discussed in this thesis. The logical progression for this work on computer

animation of modular and reconfigurable robotic systems is to extend the purely kinematic model to include the system dynamics. The inclusion of a dynamic model will allow the computer animation to be used to predict the actual behavior of the robot. There are many aspects of the extension to include the system dynamics. The dynamic equations must be generated, and they must also be solved. The dynamic model may assume rigid-body behavior, or it may be comprehensive enough to include link and joint compliances. The display may also be extended to include visual feedback concerning the system state and performance.

A purely kinematic animation of a robotic system is completely determined by specifying all of the joint displacements. An animation based on a dynamic model would display the motion of the robot given a set of joint forces and external loads. This requires that the dynamic equations be both generated and solved. Computer animation based on a generalized modular mechanical architecture is ideally suited to the automatic generation of the dynamic equations. Pertinent dynamic parameters can be stored as features within each module. As the system is graphically constructed the modules that compose the system and their connectivity, or topology, is specified. This information can then be used along with the parameters stored in each module to generate the dynamic equations for the system. If real-time performance is not required, these equations may be solved and the results stored in datafiles for animation at a later time. Solving the dynamic equations for a rigid-body, six degree of freedom robotic system requires from two to six megaflops of computer power for an update rate of from one to ten milliseconds [16]. This

level of computer performance is currently available in engineering workstations.

Extending the rigid-body dynamic animation to include joint and link deflections will require far more computer power. This may be illustrated by considering the computer generated polygonal display of a single cylinder that is approximated by thirty rectangles. Assuming that displaying a smooth bend of the cylinder will also require thirty facets along the length, the display will now require nine hundred, or n^2, polygons. For a robot animation that requires one thousand polygons for a smooth-looking rigid-body display, extension to include link and joint deformations could require one thousand squared or one million polygons for a single frame. At an animation frame rate of thirty frames per second, this would require that thirty million polygons per second be rendered by the computer.

$$30 \frac{\text{frames}}{\text{second}} * 106 \frac{\text{polygons}}{\text{frame}} = 30 * 106 \frac{\text{polygons}}{\text{second}}$$

This is beyond the current capabilities of graphics computers which peak at one million polygons per second [15]. This polygon rate requirement is also irrespective of whether the data is generated in real-time or is stored in datafiles. If real-time performance is required, it seems likely that the data will not be generated on the graphics computer. This will require that the data be sent to the graphics computer over a network. Citing the previous example of a robot scene that uses one million rectangles, it is possible to calculate the bit rate requirements for the network. Each rectangle requires four points and each point requires three parameters to be located in three-dimensional space. If each

parameter is represented as a thirty-two bit floating point number and an animation rate of thirty frames per second is desired, the network will require a bit rate capacity of at least 11.52 gigabits per second.

$$30 * 10^6 \frac{\text{polygons}}{\text{second}} * 4 \frac{\text{points}}{\text{polygon}} * 3 \frac{\text{params}}{\text{point}} * 32 \frac{\text{bits}}{\text{param}} = 11.52 * 10^9 \frac{\text{bits}}{\text{second}}$$

This is an extremely high bit rate requirement. Standard Ethernet, for example, is only ten million bits per second. These requirements seem to indicate that the animated display of robots with joint and link deformations will have to wait for further progression of computer technology.

Future development of modular and reconfigurable computer animation may also include the presentation of visual feedback concerning the system state and performance. Many different criteria could be displayed graphically alongside the animation of the robot scene. These criteria include joint forces, precision of operation, load carrying capacity, singularity approach, obstacle approach and energy consumption among others. It is also possible to present information by colored shading of the surface of the robot. Indicating stresses would seem promising for this type of display. Unfortunately, each pixel on the screen can only have one color so only one criterion at a time could be shown with surface shading. In order to present information with surface shading it is necessary to create the robot animation as a mesh. Each element in the mesh can then be assigned a color. It has, however, been shown that an animated display of a robot described as a mesh could require computer graphics performance of thirty million or more polygons per second.

Appendix 1
User's Manual

Three-dimensional computer animations of robotic systems may be created using this application program. One, two and three degree of freedom joint modules and generic links are used as the basis of this modular and reconfigurable animation package. These modules may be assembled into a large class of robotic systems that include serial, parallel, mobile and hybrid configurations. The animated model is created by specifying which modules are used and how they are connected.

The main menu bar is at the top of the screen and contains the options "FILE", "BUILD", "RUN" and "UTILITIES". These menus are of the pull-down variety and may be activated by positioning the cursor over the desired option and clicking the right mouse button. The right mouse button is used throughout this program. All lengths are in meters and all angles are in degrees.

A.1 Build

The BUILD menu is used to assemble the modules into the robotic structure. The building procedure relies upon the idea of a drawing frame. The drawing frame used in this program is Cartesian. The frame is translated and rotated in space to provide a local origin for each module. The translations are performed first and then the rotations are performed in the order x, y and then z. Each module is added to the robot model relative to this frame and may change the frame before adding the next module. The program begins with the frame in the position and orientation of the universe frame: zero x, zero y and zero z position and orientation. This x axis of the universe frame points out from the screen, the y axis goes from left to right and the z axis points up. If the view is changed this will no longer be true.

In general each module uses six arguments that may move the drawing frame before a module is added and six arguments that may move the drawing frame after the module is added. The program calls these the input frames and the output frames respectively. The modules also need to be scaled. The scaling parameters are the diameter, length, height, width, etc. With some modules the input and output frame locations are determined by the scale and the type of module used. These modules will require fewer arguments, twist angle for example, for moving the drawing frame.

Clicking the right mouse button while the cursor is positioned over the word "BUILD" in the main menu will cause the BUILD menu to appear. In this menu are the choices "add joint", "add base", "add link", "end effector",

"environment" and "parallel". Releasing the mouse button when the cursor is positioned over one of these choices will cause its associated dialog box to appear. The parallel dialogue box will ask for six arguments that will specify the position and orientation of a branching chain relative to the current position of the drawing frame. The other dialogue boxes will ask for a more specific choice of which link, joint, base, end effector or object in the environment you want to build with. Click on the specific module and then click in the OK box. Clicking in the cancel box will cause the program to return to the main menu.

A.1.1 Serial

This program assumes that modules are being added to the robot in a serial fashion unless it is told otherwise. Modules are added relative to the position and orientation of the drawing frame active when the build option was selected. The first module will be added relative to the universe frame. This module may move the frame for the next module. The input frame parameters move the drawing frame before the module is added and the output frame parameters move the drawing frame after the module is added.

A.1.2 Parallel

A parallel structure may be formed by specifying the proper joint angles to close an open serial chain. The parallel option in the menu tells the program that the current serial chain is branching. The parallel dialogue box asks for six

parameters that will specify the position and orientation of the branching frame relative to the current drawing frame. The next modules will be added as a serial chain that begins at the branching frame. The program will continue to add modules to the branching chain until an end effector is added to the chain. The end effector is a signal to the program to terminate the current chain and return to the drawing frame that was active when the parallel option was invoked. Terminating the first chain with an end effector will cause the universe frame to become active. Invoking the parallel option when the universe frame is active can be used to create multiple robots operating independently in the same environment. Attempting to create multiple chains without using the parallel option or adding more end effectors than there are chains will have unpredictable results.

A.1.3 Mobile

A mobile robot is created by attaching degrees of freedom to the base frame of the robot. There is a moving reference frame among the choices of bases. This frame takes six parameters and can be used to create a mobile robot that can move to any position and orientation in the environment. The moving reference frame can also be used to create objects that can move around in the environment, for example a pick and place operation or a falling object. Sliders and revolutes can be used to create a mobile robot whose base frame motion has less than six degrees of freedom. The parallel option can be used to create multiple mobile robots.

A.1.4 Hybrid

A hybrid robot has both parallel and serial parts. The hybrid robot can be assembled by branching into multiple serial chains and then specifying the proper joint angles to close some of the chains. A hybrid robot can also be created by adding inherently parallel modules, such as the spherical shoulder, to a serial chain. The parallel modules do not require that all of the interior joint angles be specified. The shoulder, for instance, only requires roll, pitch and yaw and then will automatically calculate the proper interior joint angles.

A.1.5 Environment

The environment menu is for placing objects in the environment. Each environment module requires six arguments that position and orient the module in relation to the universe frame. Objects added from the environment menu are fixed in their position and orientation and do not affect the active drawing frame. Available in the environment menu are the simple graphics primitives box, wedge, tube and cone. These primitives may be used to assemble more complex objects.

A.1.6 Example

This a simple example of constructing a serial three degree of freedom robot arm mounted on a fixed pedestal. Begin by running the modular cad program. A window should open with the words "FILE", "BUILD", "RUN" and "UTILITIES".

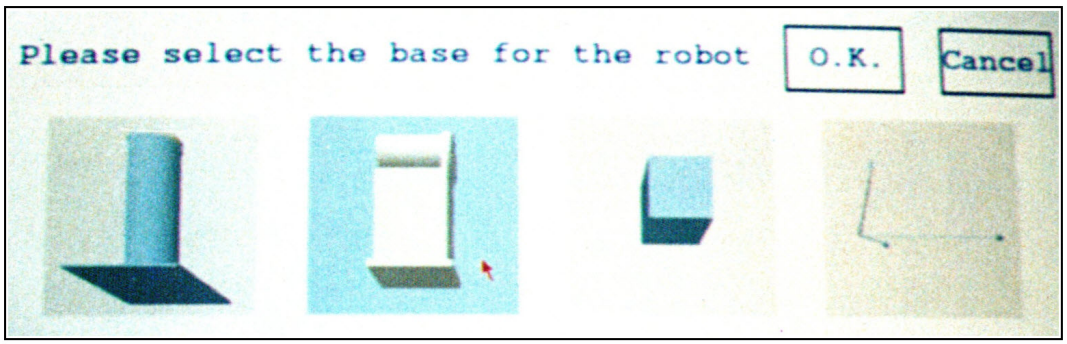


Figure A.1: Choose a base for the robot.

Select from the build menu by clicking the right mouse button when the cursor is over the word "BUILD". Look at what bases are available by releasing the button when the cursor is over "add base". Select from among the bases by positioning the cursor over the desired base and clicking the right mouse button. Select the revolute mounting pedestal and then click on the box labeled OK. A dialogue box will appear with a three-dimensional display of the revolute mounting pedestal. Choose either the slanted or the horizontal stand and the color of the pedestal by clicking on the appropriate button. The button should push in to verify the choice. Click on the OK box to add the base to the robot model

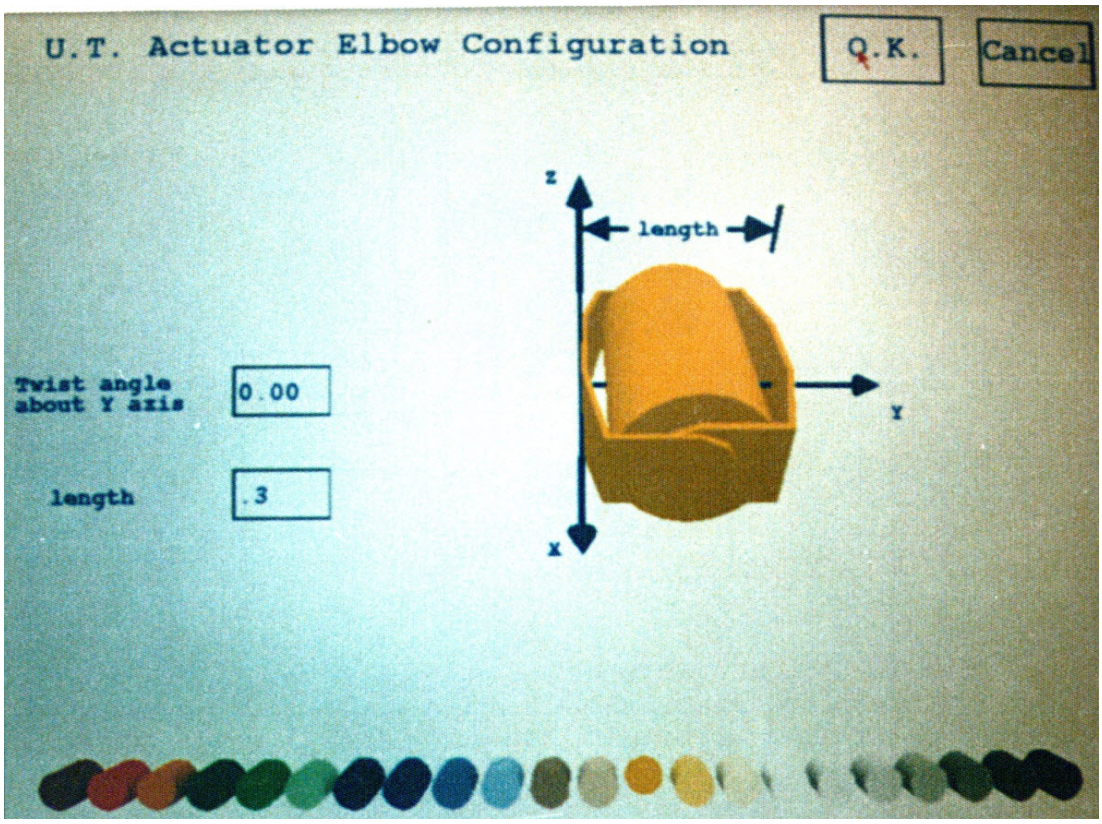


Figure A.2: Scale and select the color of the joint module.

Select the U. T. actuator from the "add joint" option in the "BUILD" menu. Scale the module by specifying the length as shown on the display. Specify the length by clicking on the box next to the word "length" in the dialogue box. Then type in the length using the keyboard. About .35 meters is fine. Choose the color, then click on the OK box. This module is now added to the robot model.

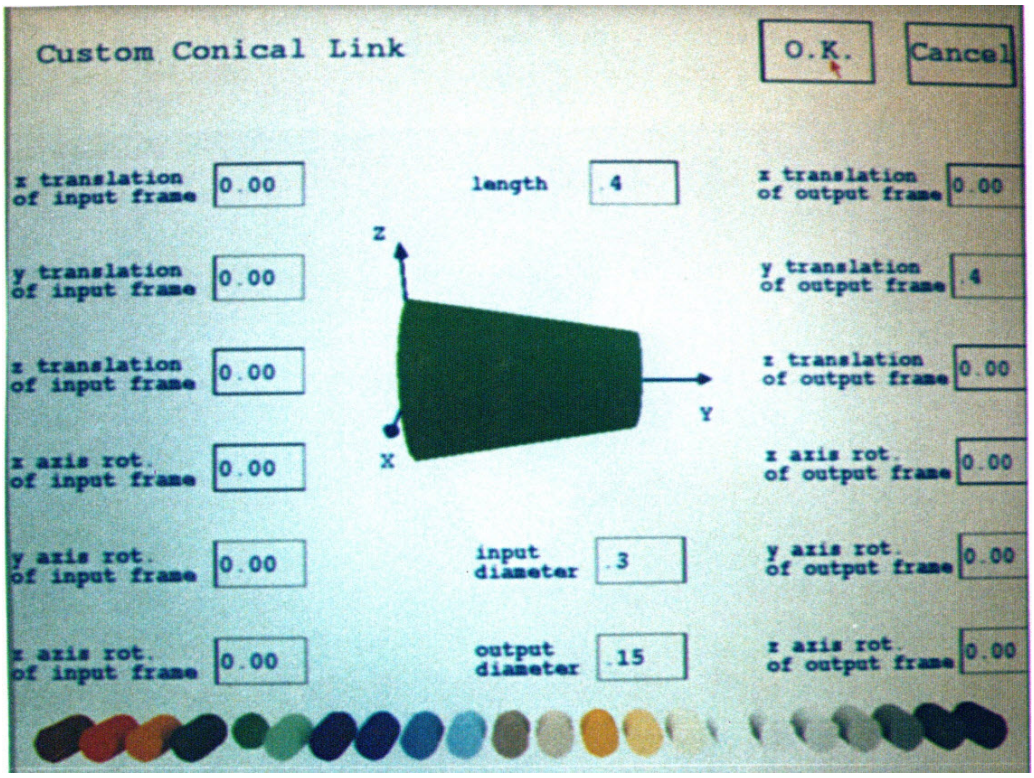


Figure A.3: Scale, color and specify the connection geometry of the link.

ow select the U. T. actuator in-line configuration from among the joints in the "add joint" option under the "BUILD" menu. Specify the two lengths shown in the dialogue box, choose the color and then click on OK. This module is now added to the robot model. Select the conical link from among the links in the "add link" option under the "BUILD" menu. Specify the length, input diameter, output diameter and color of the link. Specify the y translation of the output frame to be the same as the length so that the next module will be added onto the end of the link.

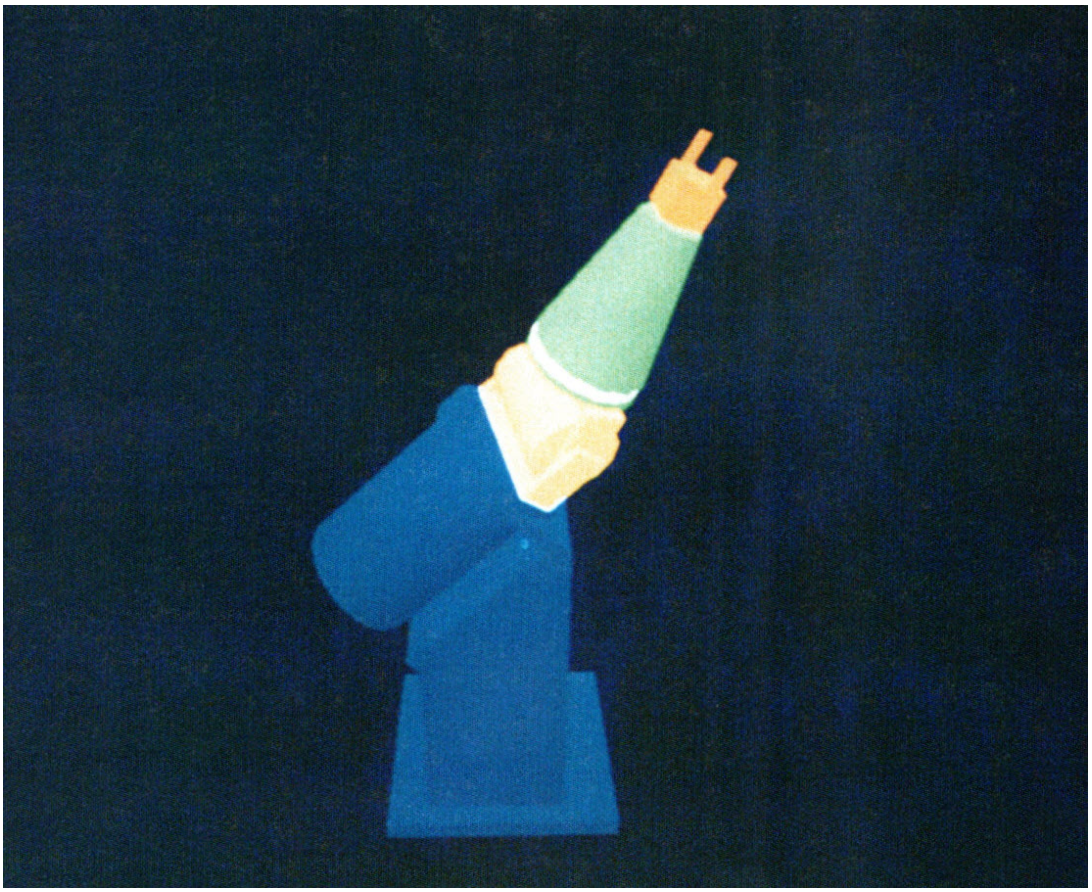


Figure A.4: Animate the finished model.

Add a parallel jaw end effector to the robot model by selecting from among the end effectors in the "end effector" option under the build menu. Animate the model by selecting "keyboard" from the "RUN" menu. Keys 1, 2, and 3 will increment and keys Q, W and E will decrement the three joints respectively.

A.2 Animation

The animation effect is created by showing rapidly in succession the robot scene with the joint angles changed by a small amount in each scene. This gives the illusion that the robot is moving. The models created with this program can be animated by accepting data from the keyboard, from data files or in real-time via shared memory.

A.2.1 Animation from Keyboard

The simplest way to animate the robot is to use the keyboard to increment or decrement each joint directly. The joints are numbered in the order that they were added to the model. The specific keys and the joints that they affect are shown in a small box that will appear near the top of the screen. Animating from the keyboard is useful for demonstrations and testing. Select "end run" from the run menu to stop animating from the keyboard.

A.2.2 Animation from Datafiles

Running the animation from a data file is an excellent method of viewing the results from inverse kinematics algorithms or dynamic simulations. To run from a datafile simply choose "datafile" in the run menu. A dialogue box will ask for the name of the datafile that is to be animated. Type in the name of the

datafile and choose whether to step or cycle through the data. If step is chosen, the computer will show the next position each time the right mouse button is clicked.

The first two lines in the datafile are reserved for comments which must be terminated by a carriage return. The next line must contain the number of degrees of freedom in the system as an integer. All of the following lines must contain a floating point number for each degree of freedom and all lines must be terminated with a return. The rotational degrees of freedom are specified in degrees and the translational degrees of freedom are specified in meters. An end-of-file token must mark the end of the file. Failing to follow this format for the data file exactly will produce unpredictable results.

A.2.3 Animation from Shared Memory

Running the program from shared memory allows the real-time animation of data as it occurs. Selecting "shared memory" from the RUN menu will cause the program to create a shared memory segment. The program will continuously read from this shared memory and display the robot in the position defined by the joint angles until "end run" is chosen in the RUN menu.

The shared memory segment is simply a one-dimensional array of floating point numbers. The program considers the first number in the array to represent the value of joint variable one, the second number joint variable two, etc. To animate the robot in real-time simply run in another window a process that is repeatedly generating joint angles and write the data into the shared

memory segment. Sample code for doing this may be found by selecting "shared memory" from the "RUN" menu and then clicking on the area marked "sample code" in the shared memory dialogue box.

A.3 Filing System

After a robot animation has been constructed, it may be saved and later recalled. The robot, objects in the environment and the viewing parameters will be saved. The joint angles are not saved and all joint angles will be set to zero when a robot model is recalled from a file.

A.3.1 Save Robot File

To save a robot model after it has been constructed simply select "save" from the "FILE" menu. Click in the name box and then type in the file name. Click OK to save the file. The program will write over any file of the same name in the current directory.

A.3.2 Open Robot File

To open a robot file simply select "open" from the "FILE" menu. Click in the name box and then type in the name of the file that is to be displayed and then click on OK. There will be an error message if the program cannot open the specified file. Opening a file will erase the current display.

A.3.3 New File

Selecting the "new" option from the "FILE" menu will display a small dialogue box. Clicking OK in this dialogue box will erase whatever is currently being displayed, return the active drawing frame to the universe position and reset the viewing parameters.

A.3.4 Quit

To end the program select "quit" from the FILE menu. The main menu bar, which is drawn into overlay planes, will not be erased if the program is not exited in this manner.

A.4 Utilities

The "UTILITIES" menu contains view, workspace and background options. The view option is used to specify the viewing parameters for the animation. The workspace option can be used to generate a graphical description of the reachable workspace of the robot. The background option sets the background color of the display.

A.4.1 View

Selecting the "view" option will cause a dialogue box to appear that can be used to adjust the viewing aspect for the scene. Note that the commands are given relative to the viewer. "Move up" means to move the viewer's eyes up, hence the scene will appear to move down. There is also an option to display the scene with orthographic or perspective viewing projections. The perspective projection allows objects to be zoomed into for a higher precision display. Orthographic projections provide higher performance.

A.4.2 Workspace

The reachable workspace is generated by cycling the robot animation through a series of positions and leaving a token at the end effector for each position. The token is a triangle whose normal points along the y axis of the last module in the chain. The workspace dialogue box will ask for a range of motion and a step size for each degree of freedom. The color buttons specify the color of the triangular token. The workspace generation is quite computationally intensive and will take a long time if large ranges are used with small step sizes. The program will indicate the approximate amount of time in minutes that the calculation should be expected to take.

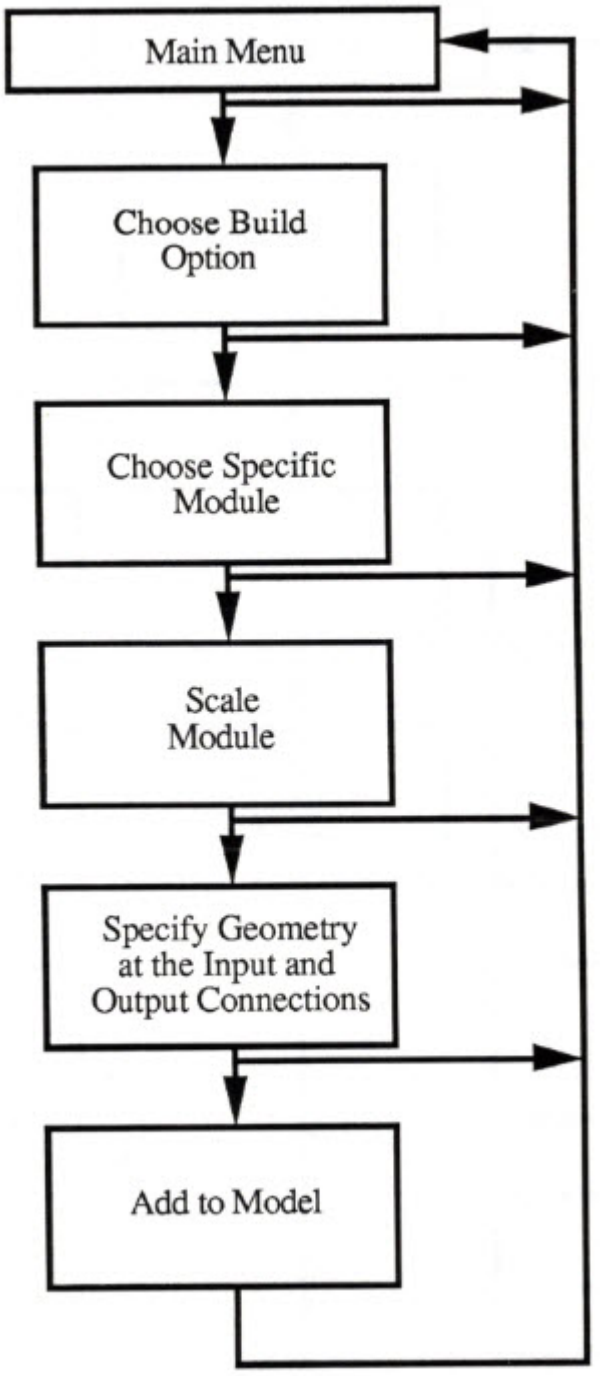


Figure A.5: Flow of control for build option.

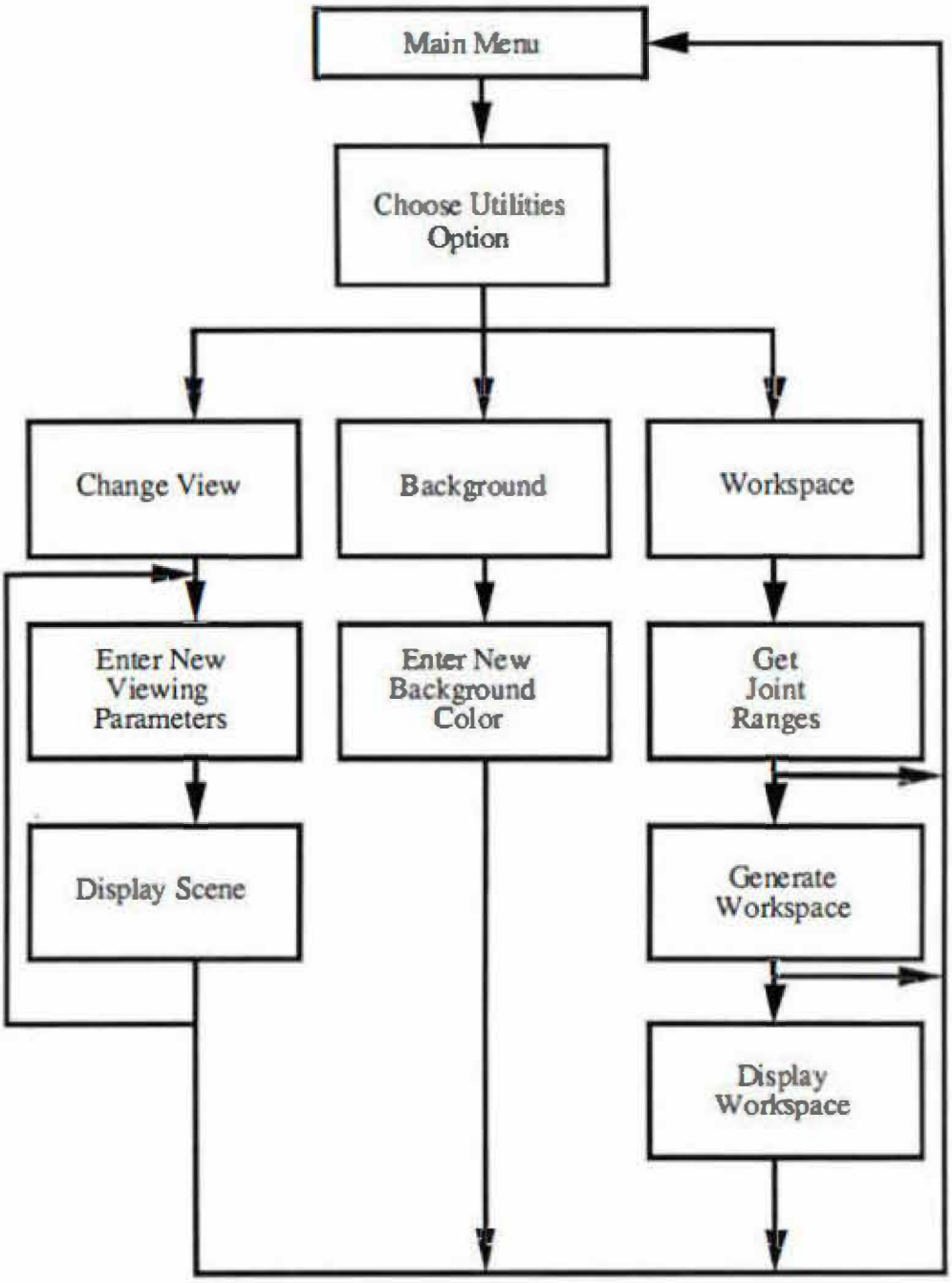


Figure A.6: Flow of control for utilities option.

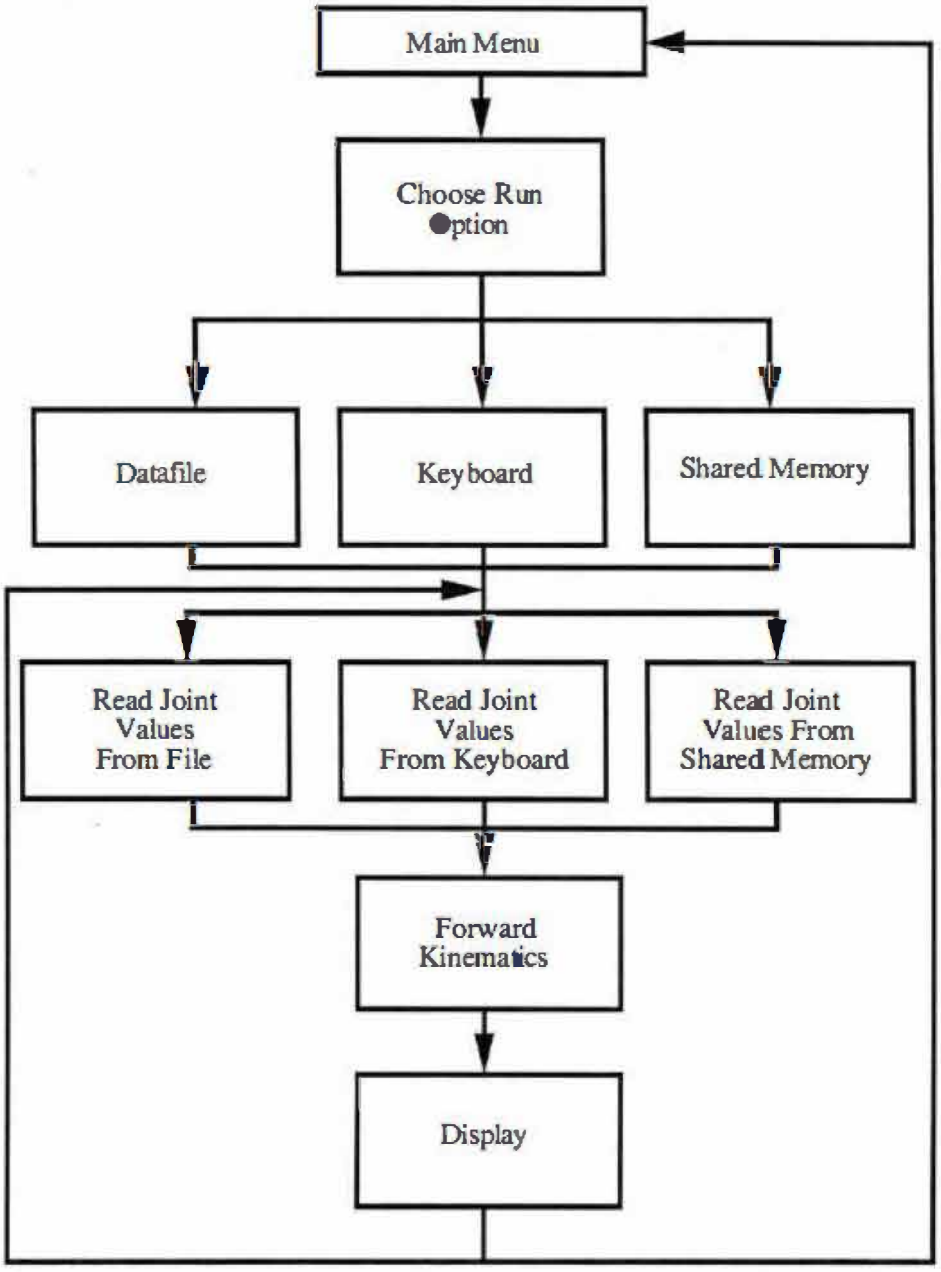


Figure A.7: Flow of control for run option.

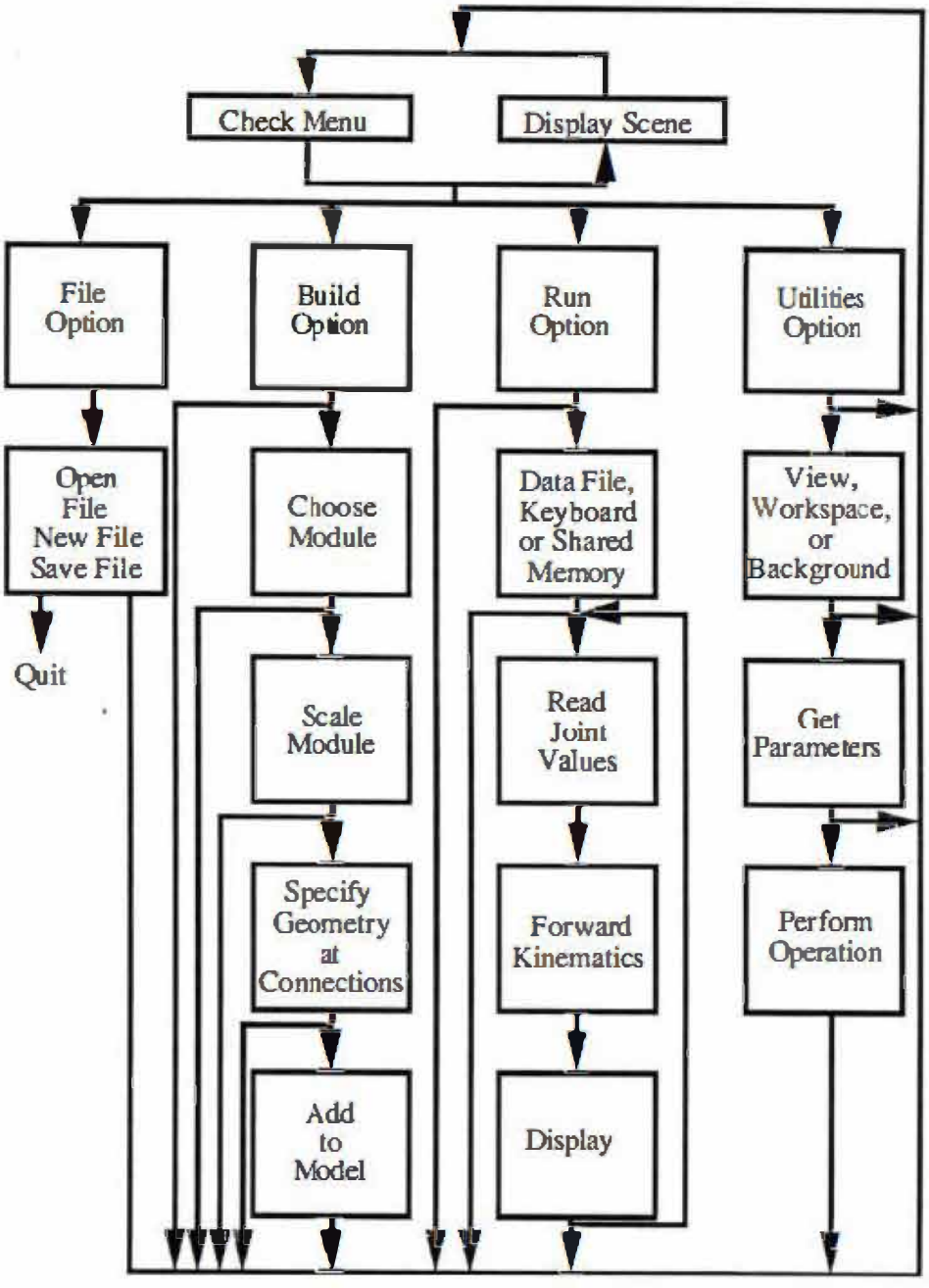


Figure A.8: Flow of control for main loop.

Bibliography

[1] Ambler, A. P. "Robotics and Solid Modelling: A discussion of the Requirements Robotics Applications put on Solid Modelling." Robotics Research, The Second International Symposium. 1985. pp. 361-367.

[2] Bean, R. W. and R. L. Norton. "Adaptive Robotic Fine Positioning System." *ASME Computers in Engineering*. Vol. 1. 1986. pp. 113-119.

[3] Boren, R. R. "Graphics Simulation and Programming for Robotic Workcell Design." *Robotics Age*. August, 1985. pp. 30-33.

[4] Cho, W., D. Tesar and R. Freeman. "The Dynamic and Stiffness Modeling of General Robotic Manipulator Systems with Antagonistic Actuation." IEEE International Conference on Robotics and Automation. Vol. 3. 1989. pp. 1380-1387.

[5] Cleary, K. and D. Tesar. "Incorporating Multiple Criteria in the Operation of Redundant Manipulators." IEEE International Conference on Robotics and Automation. Vol. 1. 1990.

[6] Cox, D. J. and D. Tesar. "The Dynamic Model of a Three Degree of Freedom Parallel Robotic Shoulder Module." Proceedings of The 4th International Conference on Advanced Robotics. 1989.

[7] Crane III, C. D. and J. Duffy. "An Interactive Animated Display of Man-Controlled and Autonomous Robots." *ASME Computers in Engineering*. Vol. 1. 1986. pp. 35-38.

[8] Crowley, J. L. "World Modelling and Position Estimation for a Mobile Robot Using Ultrasonic Ranging." IEEE International Conference on Robotics and Automation. Vol. 2. 1989. pp. 674-680.

[9] Dooner, M. "Robotics Software and CADCAM." *Journal of Computer Aided Engineering*. 1984. pp. 217-220.

[10] Euler, J. A., R. V. Dubey, S. M. Babcock and W. R. Hamel. "A Comparison of Two Real-Time Control Schemes for Redundant Manipulators with Bounded Joint Velocities." IEEE International Conference on Robotics and Automation. Vol. 1. 1989. pp. 106-112.

[11] Fernandez, K. and G. E. Cook. "Use of Computer Graphic Simulation Techniques for Robot Control System Development." Proceedings of the 18th Southeastern Symposium on System Theory. 1986. pp. 433-441.

[12] Flora, P. C. *International Robotics Industry Directory*. 4th. ed. Conroe, TX. Technical Database Corp. 1986.

[13] Freund, E. and H. Hoyer. "Collision Avoidance in Multi-Robot Systems." Robotics Research, The Second International Symposium. 1985. pp. 135-146.

[14] Fujimura, K. and H. Samet. "Time-Minimal Paths Among Moving Obstacles." IEEE International Conference on Robotics and Automation. Vol. 2. 1989. pp. 1110-1115.

[15] Goren, K. "Periodic Table of the Irises." Silicon Graphics Computer Systems. May, 1990.

[16] Hernandez, E., D. Tesar and R. Sreedhar. "Computational Requirements for the Design and Control of the Fifth Generation Robot." Report to Cray Research and the U. S. Department of Energy. September 1989.

[17] Hemani, A. "Joint Velocity in Redundant Robot Manipulators." *Robotica*. Vol. 8. 1990. pp. 69-72.

[18] Hirose, S. and S. Ma. "Redundancy Decomposition Control for Multi-Joint Manipulator." IEEE Interantional Conference on Robotics and Automation. Vol. 1. 1989. pp. 119-124.

[19] Holcombe, L., R. Larson and M. Montemerlo. "Overview of the NASA Automation and Robotics Research Program." AIAA/NASA Symposium on Automation, Robotics and Advanced Computing for the National Space Program. 1985.

[20] Hudgens, J. C. and D. Tesar. "A Fully-Parallel Six Degree-of-Freedom Micromanipulator: Kinematic Analysis and Dynamic Model." Proceedings of the 20th ASME Mechanisms Conference. 1988. pp. 29-37.

[21] Kang, H. and R. A. Freeman. "An Interactive Software Package (MAP) for the Dynamic Modeling and Simulation of Parallel Robotic Systems Including Redundancy." *ASME Computers in Engineering*. 1990. 117-124.

[22] Kim, W. S. and L. W. Stark. "Cooperative Control of Visual Displays for Telemanipulation." IEEE International Conference on Robotic and Automation. Vol. 3. 1989. pp. 1327-1332.

[23] Klein, C. A. "Use of Redundancy in the Design of Robotic Systems." 2nd. International Symposium of Robotics Research. 1984. pp. 58-65.

[24] Kovacs, W. "Previewing Robotics Motion with Computer Graphics." *Robotics Age*. August, 1985. pp. 16-19.

[25] Lee, S. "Dual Redundant Arm Configuration Optimization with Task-Oriented Dual Arm Manipulability." *IEEE Tranactions on Robotics and Automation*. Vol. 5, No. 1, February 1989. pp. 78-97.

[26] Levas, A. and R. Jayaraman. "WADE: An Object Oriented Environment for Modeling and Simulation of Workcell Applications." *IEEE Tranactions on Robotics and Automation.* Vol. 5, No. 3, June 1989. pp. 324-335.

[27] Liubinskas, A. and R. Priemer. "A Programming Language for Real-Time Control." *ASME Computers in Engineering.* Vol. 1. 1986. pp. 43-46.

[28] Meckel, P. H. and W. P. Seering. "Feedforward Control Techniques to Achieve Fast Settling Time in Robots." American Control Conference. Vol. 3. 1986. pp. 1913-1918.

[29] Mirolo, C. and E. Pagello. "A Solid Modelling System for Robot Action Planning." IEEE Computer Graphics and Applications. Vol. 8. 1988. pp. 55-67.

[30] Mussa-Ivaldi, F. A. and N. Hogan. "Solving Kinematic Redundancy with Impedance Control: A Class of Integrable Pseudoinverses." IEEE International Conference on Robotics and Automation. Vol. 1. 1989. pp. 283-288.

[31] Noborio, H., T. Naniwa and S. Arimoto. "A Feasible Motion-Planning Algorithm for a Mobile Robot on a Quadtree Representation." IEEE International Conference on Robotics and Automation. Vol. 1. 1989. 327-332.

[32] O'Neil, S. R. and M. Jamshidi. "ROBOT_S: An Interactive Robot Simualtion Language." *Robotics*. Vol. 4. 1988. 245-256.

[33] Ouh-young, M., D. V. Beard and F. P. Brooks Jr. "Force Display Performs Better than Visual Display in a Simple 6-D Docking Task." IEEE International Conference on Robotics and Automation. Vol. 3. 1989. pp. 1462-1466.

[34] Parker, J. K., D. E. Goldberg and A. R. Khoogar. "Inverse Kinematics of Redundant Robots using Genetic Algorithms." IEEE International Conference on Robotics and Automation. Vol. 1. 1989. pp. 271-276.

[35] Pillon, G. "OLP: The Key to Expanding Robot Usage." *The Industrial Robot*. Vol 15. 1986. pp. 206-210.

[36] Pocock, G. "A Distributed, Real-Time Programming Language for Robotics." IEEE International Conference on Robotics and Automation. Vol. 2. 1989. pp. 1010-1015.

[37] Salcudean, S. and C. An. "On the Control of Coarse-Fine Manipulators.". IEEE International Conference on Robotics and Automation. Vol. 3. 1989. pp. 1834-1840.

[38] Salisbury, J. K. and J. T. Craig. "Articulated Hands: Force Control and Kinematic Issues." *International Journal of Robotics Research*. Vol. 1, No. 1. 1982. pp. 4-17.

[39] Salkind, L. "The SAGE Operating System." IEEE International Conference on Robotics and Automation. Vol. 2. 1989. pp. 860-865.

[40] Schmitz, D., P. Khosia, R. Hoffman and T. Kanade. "CHIMERA: A Real-time Programming Environment." IEEE International Conference on Robotics and Automation. Vol. 2. 1989. 846-852.

[41] Sheridan, T. B. "Human Supervisory Control of Robot Systems." IEEE Conference on Robotics and Automation. 1986.808-812.

[42] Shulman, S. "When a Nuclear Reactor Dies, $98 Million is a Cheap Funeral." *Smithsonian*. Vol. 2. October, 1990.

[43] Simon, R. L. "The Marriage Between CAD/CAM Systems and Robotics." *Design News*. Vol. 39. 1983. pp. 87-94.

[44] Technomatix. "Four Steps to Simulating Robotic Workcells." *Robotics Engineering*. August, 1986. pp. 23-25.

[45] Tesar, D. "An Assessment of the Development and Application Potential for Robots to Support Space Station Operations." AIA/NASA/AF Symposium for Automation and Robotics for Space Operations. 1985

[46] Tesar, D. and M. Butler. "A Generalized Modular Architecture for Robot Structrues." *Manufacturing Review*. Vol. 2. June, 1989. pp. 91-117.

[47] Tesar, D., D. Sreevijayan and C. Price. "Four Level Fault Tolerance in Manipulator Design for Space Operations." First International Symposium on Measurement and Control in Robotics. 1990.

[48] Thomas, M. and D. Tesar. "Dynamic Modelling of Serial Manipulator Arms." *Transactions of the ASME*. Vol. 104. September, 1982. pp. 218-228.

[49] Thomson, C. and H. R. H. Holt. "Solid Modelling and Robot Simulation - Kinematics for Off-Line Programming." *U. K. Robotics Research*. 1984. pp. 93-98.

[50] Wander, J. and D. Tesar. "Real-Time Computation of Influence Coefficient Based Dynamic Modelling Matrices for Improved Manipulator Control." Masters Thesis; University of Texas. Austin, Texas. 1985.

[51] Wang, K. and T. K. Lien. "The Planning of a Straight Line Trajectory via Interactive Computer Graphics." *Robotics & Computer-Integrated Manufacturing*. Vol. 5. 1989. pp. 215-221.

[52] Yi, B. J., R. A. Freeman and D. Tesar. "Open-Loop Stiffness Control of Overconstrained Mechanisms/Robotic Linkage Systems." IEEE International Conference on Robotics and Automation. Vol. 3. 1989. pp. 1340-1345.

[53] Yoshikawa, T. "Manipulability of Robotic Mechanisms." *International Journal of Robotics Research*. Vol. 4, No. 2. 1985. pp. 3-9.

VITA

Richard Nelson Hooper was born on December 27, 1962 in Amarillo, Texas. His parents are Howard Thomas Hooper and Carol Lee Hooper. Richard graduated from William Howard Taft High School in Woodland Hills, California in May of 1981. He graduated from Rice University in Houston, Texas in May of 1985 with the degree of Bachelor of Science in Electrical Engineering. After working as an electrical engineer for several years he entered the Graduate School of the University of Texas in September of 1988. Richard married the lovely Mary Carrington Foye on December 30, 1988 in Houston, Texas.

Permanent Address: 1709 Barn Swallow
Austin, Texas 78746

This thesis was typed by Richard Nelson Hooper.