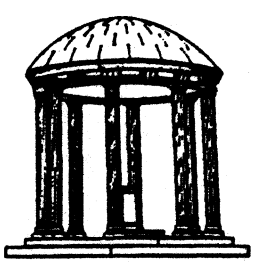
Force Display In Molecular Docking

TR90-004

February, 1990

Ming Ouh-young



The University of North Carolina at Chapel Hill
Department of Computer Science
CB#3175, Sitterson Hall
Chapel Hill, NC 27599-3175

FORCE DISPLAY IN MOLECULAR DOCKING

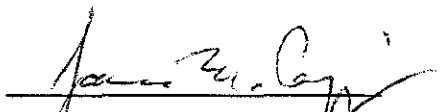
by
Ming Ouh-young

A Dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
1990

Approved by:

Advisor: Frederick P. Brooks



Reader: James Coggins

Reader: David Beard

Acknowledgement

I would like to thank my thesis advisor, Professor Frederick P. Brooks, Jr., for his full support of the ARM project. In 1976 he already had the idea to the use force-feedback ARM in the molecular docking problems. Dr. Mike Cory and Professor David Richardson were the pioneer users of the prototype system from the beginning.

Professors James Coggins, David Beard, and Victor Klymenko have given useful suggestions in my experimental design.

The ARM project is a team work, many people contributed to the development of the ARM system. Michael Pique, former GRIP director, laid out all the necessary components of the ARM system, even before I joined the GRIP project. Successive GRIP director Helga Thorvaldsdottir and GRIP chemist Mark Harris helped me with software and chemical problems. John Hughes has been in the ARM team since the beginning of the project; he built most of the hardware components, and maintained them. Neela Srinivasan wrote the PS300 interface, and the second version of the energy evaluator. Ji-Fang Wang helped me to establish the link to Pixel-planes-4 graphics engine.

This research was supported by the NIH Division of Research Resources, RR-02170. Dr. Lee Kuyper and Dr. Mike Cory furnished the molecular data. There were twelve biochemists participating in my molecular docking experiments. They had provided useful advices in improving the system design. They are Hope Taylor, Dora Schnur, Joe Chan, Amil Anderson, Richard Lombardy, Paul Charifson, Kim Gernert, Neil Tweedy, Aron Miller, Tom Quinn, Michael Hecht, and Neela Srinivasan.

Contents

List of Figures

List of Tables

Chapter 1

Using a Manipulator for Force Display

A person manipulating real objects ordinarily sees the objects and feels reactive forces. Working remotely (for example, in space or under the sea) or in virtual worlds (for example, in molecular models), the operator lacks direct force feedback. Under such conditions, the system designer can either do without force information or deliver synthesized forces.

We have developed a real-time molecular-docking system that uses an electrically coupled remote manipulator as a force display. Integrated with interactive computer graphics and a high-speed calculation of the interaction forces between a drug and a receptor site in a molecule, the system is designed to be a tool for molecular scientists. In our system, the manipulator displays the forces and torques proportional to those exerted on the drug as the drug molecule is juxtaposed to the receptor site by the user's hand. The manipulator serves both as an input device for 6-D manipulation, and as an output device for generating forces.

Considered as conceptual descendants of force-feedback remote manipulators, GROPE-III (the name of our system) and a conventional robot are opposites. A conventional robot is a master-slave remote manipulator with a computer model assuming the role of the master station and its user. Grope-III is a remote manipulator with a computer model assuming the role of the slave station and its task-world.

1.1 Introduction to the application

Scientific visualization. Visualization of scientific data sets plays an important role in understanding complex phenomena. We say that scientists *grasp* new understandings, that they acquire a *feel* for the behavior of computed models. In this sense, a force display would

be a natural tool to provide a *feel* of the computed models.

Force display is not just another technology. As an integrated part of a scientific visualization, a force display can serve both as a tool for discovery and understanding, and as a tool for communication and teaching. According to the visualization report, "We speak (and hear) — and for 5000 years have preserved our words. But, we cannot share vision" [McCormick 87]. To this, I would add "we cannot share vision and feel."

Can a force display help in scientific visualization? We believe it can, so we chose a particular application, molecular docking, for our study.

1.1.1 The molecular docking problem

The basis of the biological activity of many drug molecules is found in their mode of interaction with a specific receptor site on a protein or nucleic acid molecule. The detection of allowable and forbidden dockings is crucial to analytic drug design and very important for understanding the action of toxins, antibiotics, and carcinogens.

Definition. The major concern of molecular docking problems is to find the position and orientation of a drug with respect to a receptor site, such that the interaction energy is a global minimum. Goodness of fit (geometric fit, electrostatic fit, hydrogen-bond fit, etc.) is often used as a criterion to describe the molecular docking problem qualitatively. Even when the molecules are approximated as rigid bodies, the problem is difficult, because the proteins usually contain hundreds to thousands of atoms, the drug has at least six degrees of freedom, and the surfaces are irregular, so that local energy minima abound.

1.2 Thesis statement

Algorithmic search of the configuration space is extremely costly: a simple docking problem takes hours even on a fast machine. The investigator can use his knowledge of chemistry to do a far more efficient job of pruning and directing the search. We investigate the conjecture that knowledge-guided searching by a human expert would take less total time than more ignorant searching by a fast computer. My theses are

> The addition of force display to an interactive computer graphics system can significantly help in molecular docking problems in terms of task-completion time. Molecular scientists can direct the fitting of the drug molecule in a receptor

molecule so that interaction energy is in the neighborhood of the true global energy minimum, given assistance such as visual and force display.

The task-completion time is the period to reach the neighborhood of the true global energy minimum. Once in the neighborhood of the true global energy minimum, any gradient-based method can converge to the minimum efficiently. Impossible drug-receptor configurations are immediately self-evident, just as a wrong key does not fit a lock. The system is designed to accommodate different force models for molecular interaction. It can therefore be applied to other virtual-world manipulations, if one has a force model of the virtual world.

1.3 Force display in addition to visual display

During the past two decades, X-ray crystallographic studies have determined the three-dimensional structures of some drug-receptor complexes. Recently, real-time interactive computer graphics has contributed to the understanding of interactions of molecules, using various techniques such as solvent-accessible surfaces and space-filling van der Waals spherical models. The main focus has usually been on geometric configurations, i.e., van der Waals forces. But geometric fit does not describe the drug-receptor interaction completely. Most drugs have partial atomic charges in electrostatic interaction with charges distributed in the receptors.

If we ask how we would like to see the docking of two complicated 3-D surfaces against each other, it is hard even to imagine a satisfactory visualization. What is really wanted is a display in which we can move models of the molecules while both seeing them and feeling all the forces and torques. Sutherland first put forth this vision in 1965 [Sutherland 65].

Guided by the above concept, I have constructed a force-display system to allow molecular scientists to feel and to see the goodness of geometric and electrostatic fit between two molecules. I chose to use an Argonne E-3 Remote Manipulator (ARM), built by Ray Goertz and colleagues, as my force-output device [Goertz 61].

Since our system is designed to furnish a dynamic experience of the virtual world of the molecules, the following problems must be solved in real time. First, how can the user naturally control the drug motion and its conformation (shape)? Second, how can we display the molecules to get a good visualization in real time? Third, how can we quickly calculate the forces between the drug and the protein? Finally, how can we synthesize and present to the user the forces and torques exerted on the drug molecule?

1.4 The ARM system design

This section briefly describes the system design and the current system configuration. More detailed hardware and software descriptions are in chapter 10 and Appendices 1 and 3.

1.4.1 System configuration

The GROPE-III molecular modeling system (Figure 1.1) consists of the Argonne E-3 Remote Manipulator (ARM) for force and torque presentation to the user, a graphic system (Evans & Sutherland Picture System PS300 or Fuchs's Pixel-Planes, a very fast raster graphics engine developed at UNC-Chapel Hill) for displaying proteins and drugs, a Tektronix alternating polarization plate for stereo images, and software for energy and force calculations. The system operates under Unix, with Ethernet connections among the graphic systems and a SUN4 workstation, whose digital-to-analog (D-A) and analog-to-digital (A-D) converters control the ARM, and which maintains the virtual model. At present we achieve 22 to 30 updates/second in force simulation for real molecules, and 15 Hz in updating stereo images.

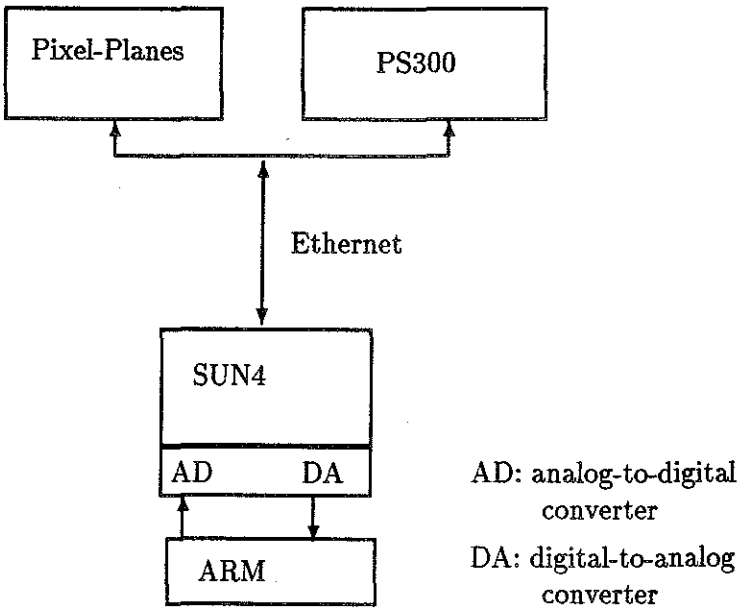
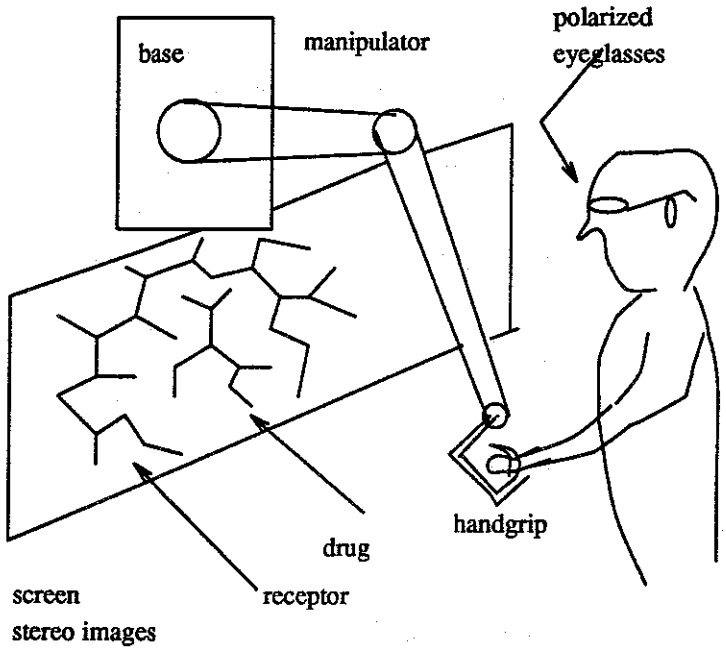


Figure 1.1: System configuration.

Figures 1.2 and 1.3 show the system diagram and a photograph of the user interfaces. The screen displays the virtual molecular world in stereo. The interaction forces and torques among molecules are based on a molecular mechanics model. If the user wants to move the drug in 6-D or to change the viewpoint, he can hold and move the handgrip of the manipulator as a 6-D joystick. At the same time, the user can feel the forces and torques through the manipulator as if he were holding the drug.

The user can control the output force by turning a dial, and he can turn off the force by using a foot switch. To model the drug molecule, the user can (1) use a set of dials to control the torsional angles, and (2) activate or deactivate subgroups of the drug by keyboard commands.



The molecular scientist is using the system to dock the drug into the receptor. The system has visual and force display at the same time.

Figure 1.2: The molecular docking system

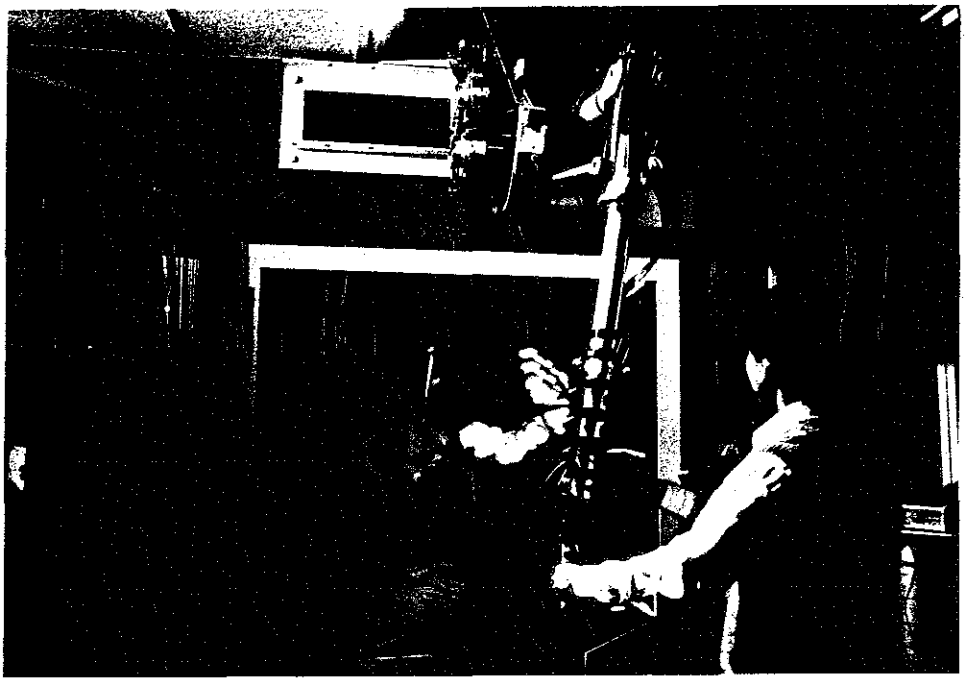


Figure 1.3: A photograph of the system

1.5 Real-time molecular force and energy calculation

The energy of interaction between a drug and its receptor molecule can be approximated as the sum of electrostatic (Coulombic) and van der Waals interaction energies using the Lennard-Jones 6-12 potential function as follows [Pattabiraman 85]:

$$Vtot = \sum_{d=1}^{M}\sum_{r=1}^{N}\{\frac{332 * q(d) * q(r)}{k * R(r,d)} + [\frac{A(r) * A(d)}{R(r,d)^{12}} - \frac{B(r) * B(d)}{R(r,d)^{6}}]\}$$

where $Vtot$ is the total energy of interaction, M is the number of atoms in the drug molecule, N is the number of atoms in the receptor molecule, $q(r)$ and $q(d)$ are the charges of the atoms in the receptor and the drug molecules respectively, R(r,d) is the distance between the drug atom d and the receptor atom r; $k, A(r), A(d), B(r), B(d)$ are the dielectric and non-bonded constants. In the above equation, the first term corresponds to the electrostatic interaction and the second and the third terms to the repulsive and attractive terms in the van der Waals interaction energy.

In AMBER, a molecular modeling system, there is a correction term devoted to hydrogen bonds, because the hydrogen bond will be shorter if no correction is made [AMBER 80]. I

did not implement it in the GROPE-III, partly because the linear interpolation on tabulation data tends to increase the hydrogen-bond length, and so the correction term is sometimes unnecessary. For bond rotations, I use the same model above; however, there is one torsional term that defines the energy variation as a function of the rotation angles. This parameter can be obtained from the AMBER package [Weiner and Kollman 84].

1.5.1 Fast force calculation— Grid evaluation and linear interpolation

Pattabiraman developed a 3-D grid tabulation for real-time calculation of the total interaction energy between a drug and its receptor molecule [Pattabiraman *et al.* 80]. If the size of the drug is M atoms, and the size of its receptor is N atoms, direct evaluation of the total energy has a time complexity of $O(M * N)$. Using the grid method, the potentials applied to each grid point (suppose there is an atom with unit charge and unit van der Waals radius here) by all the receptor atoms are pre-calculated and stored. When the drug is moved in the grid space, each atom in the drug can find a closest grid point and from the table find its contribution to the total energy, with the precision related to the grid spacing. Now, the problem becomes a $O(M)$ complexity problem. Usually, N is of the order of a few thousand, whereas the drug size M is only of the order of a few dozen. This is a great improvement in computation time, at the cost of much storage for grid tabulation. We tabulate interatomic forces similarly, except that vector quantities must be tabulated and vector operations must be used for force components.

For force output to the user, the discrete forces from the grid evaluation will be perceived as bumpy. Linear interpolation is thus used to make the force function continuous.

1.6 Modifications to the Argonne E-3 manipulator

We made several modifications to the Argonne E-3 manipulator. First, we removed the slave arm. The original synchro transformer pair was isolated and reconfigured to sense position. The result is an amplitude-modulated voltage from the synchro transformer. A peak detector is used to demodulate this signal to get the synchro transformer shaft position.

Second, since we do not use the gripping degree of freedom in the original ARM, we replaced the original handgrip with a hand-control previously used for radar-positioning (Figure 1.4). We assigned a switch that can be toggled to change between two control modes— the viewpoint control and the drug-molecule control. We assigned the trigger in the hand-control to serve as a clutch to allow the user to move the manipulator without moving the objects. This is important because the manipulator has a limited working volume, and

the virtual world is boundless. We added a force-scaling dial to the hand-control to allow the user to adjust the force range. Finally, we added a set of 10 dials to control chemical-bond rotations. Force is not directly displayed (fedback) to the fingers as they rotate these bonds.

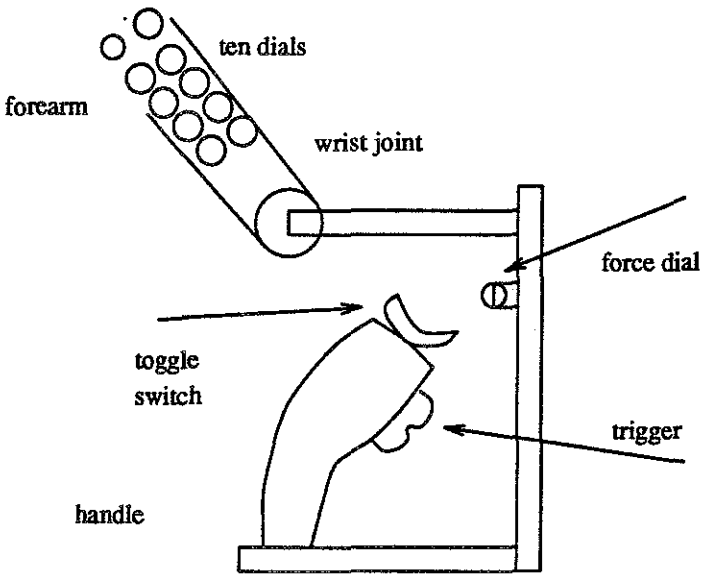


Figure 1.4: The side view of a modified hand-control at the wrist

1.7 Jacobian matrix, force control, and transformations in different coordinate systems

To generate the desired forces and torques through the manipulator, a Jacobian matrix can be used to calculate the torques and forces at each joint [Paul 80]:

$$T = J^{transpose} * F \tag{1.1}$$

Force $F = [Fx, Fy, Fz, Mx, My, Mz]$, joint torque $T = [T1, T2, T3, T4, T5, T6]$, (Fx, Fy, Fz) is a force vector, and (Mx, My, Mz) is a torque vector. The Jacobian matrix J is a function of the manipulator joint angles. Appendix 1 gives the detailed derivations of the ARM Jacobian matrix.

We have developed a useful software tool that can take descriptions of each joint in a manipulator and generate the associated matrix for that joint, symbolically manipulate the matrices, and generate a Fortran or C routine for the Jacobian matrix calculation.

Perceived forces and moments change as coordinate frames change. In our system, the problem is complicated because the user can change viewpoint on the fly. We had considerable difficulties just verifying the nested force and moment transformations. For example, to trace the forces in the system, we have the following variables to reflect the changing coordinate systems: **force-in-molecule (force-in-grid), force-in-armbase, force-in-handgrip, force-in-screen.** Moments are similarly named. The drug can be manipulated in 6-D as if we held it at a centered pivot point. The viewpoint is changed by rotating the entire drug-receptor complex about the receptor-molecule center. Visual display has proved very helpful in getting the transformations correct.

1.8 The hard surface problem

Kilpatrick and other researchers have always found it is extremely difficult to simulate hard-surface bumping forces realistically. When a human bumps into a real table-top, he feels it suddenly and almost instantly. That is not the case in our system. The Coulomb force is easy to synthesize, but we have trouble simulating the van der Waals hard-surface forces. Therefore, a bump checker was developed to augment bumping display. When one atom bumps into another, a flashing vector appears between the two on the screen. We thus create a simulated bump condition in the user's mind, using visual cues and force feedback at the same time. This proves to be useful in geometric docking.

In section 6.4.2, "Hard-Surface Simulation with Low Sampling Frequency," I describe another way of dealing with the hard-surface problem by taking advantage of control theory.

1.9 Summary of results

My work focuses on the generation and presentation of forces and torques, the experimental design and evaluation of force display in real molecular docking problems, and the theory and analysis of creating an illusion of feel.

What I have accomplished in my dissertation work is:

1. Implementation of a real-time molecular docking system.

2. Analysis of limitations and constraints and corresponding solutions to the general docking problem by using a force-feedback device. I derived several rules which can explain and guide the solution of many problems in force-feedback systems.

3. Evaluation of the performance of the 6-D force-feedback ARM as a tool in docking within a force field consisting of simulated springs. The performance is defined as the task time to get the system potential energy to fall below a certain threshold. The results of this simplified docking problem showed

   (a) Performance with a force display is better ($p < 0.01$) than performance with a visual display alone.

   (b) Subjects are able to find the zero force position more than twice as fast with a force display alone than with a visual display alone.

   I also describe a new way of graphically representing the resultants of a set of forces and torques acting on a body. Even though the experiment shows force display to be more effective, it also shows that the simple 6-D docking task can reliably be done with this visual display alone.

4. One controlled experiment plus case-by-case study to evaluate the usefulness of force display in the application of molecular docking. Twelve biochemists participated in the experiment with real research molecules. The experimental results corroborated ($p < 0.05$) my hypothesis in 6-D rigid-body manipulation. There was, however, no significant difference in the case of one-degree-of-freedom chemical-bond rotations. That is, for this 1-D task, the visual cues sufficed. Limited case-by-case studies show that the subjects using the current ARM system are doing faster and getting more precise docking results than those using the Ellipsoid algorithm (one of the best batch-computed docking algorithms to date), both in the number of well-formed hydrogen bonds and in displacements.

1.10 Overview

Chapter 2, "A Survey of Force Display Systems and Docking Tools," surveys two categories of systems: the force-feedback systems and the molecular modeling tools.

Chapter 3, "A Scenario of Molecular Docking," describes what molecular docking is like when using the GROPE-III system. The detailed functions of the GROPE-III system are given here.

Chapter 4, "Force Display Performs Better than Visual Display in a Simple 6-D Docking Task," describes a simple experiment with six virtual springs. Seven subjects tried to find the null-force position.

As a byproduct of this work, I developed a new method of docking using visual display of forces and torques and no kinesthetic display. This method can be applied to workstations.

Chapter 5, "A Molecular Docking Experiment," is the major result of my thesis. The experiment simulated some of the interaction forces between the dihydrofolate reductase enzyme (600 atoms) and six drugs (40 to 60 atoms each). Twelve biochemists tried to dock the drugs into the enzyme. Because real research problems are used in the experiment, design considerations are complicated. Besides the statistical findings, I observed some interesting phenomena.

Chapter 6, "Creating an Illusion of Feel: Control Issues in Force Display," deals with the theory and analysis of a force-display system. Without such an analysis, one cannot answer the following basic question: What is the required system-updating rate for system A doing simulation B? Many puzzles are solved using the analysis, including the hard surface contact problem.

Chapter 7, "Determining the ARM's Mechanical Impedance," describes one convenient way to measure the ARM's parameters, such as damping and inertia, as an application of the theories in Chapter 5. Similarly, the parameters of the human arm and of Minsky's force-feedback joystick are determined. These parameters are important for force-feedback-system designers.

Chapter 8, "Algorithms vs. Mind-Guided Exploration," describes my experiences in molecular docking using algorithms. This chapter shows why a mind-guided exploration of the space of possible configuration might have an advantage.

Chapter 9, "Software Constructs," describes the software modules that are necessary in doing a simulation.

Chapter 10, "System Configuration and Hardware Design," describes the hardware configurations. The focus is on the various design considerations to speed up the system. I describe three generations of the ARM system, together with their performances.

Chapter 11, "Contribution and Future Work," summarizes my contribution and recommends future directions.

Appendix 1, "The Jacobian Matrix of the ARM," describes one way of deriving the ARM's Jacobian matrix. It also contains formulae that tell how to transform forces and moments among coordinate frames. Appendix 2, "Preparing Test Molecules for Molecular Docking," describes in detail how to prepare internal files for docking, if we want to model a set of molecules. Appendix 3, "Control Routines for the ARM," gives the specifications of important library routines of the ARM. If one wants to write his own application using the ARM, this is the document to use. Appendix 4, "Running an ARM Demo," describes how to run the ARM demo programs. All the user needs to do to run a demo program is to follow the given instructions.

1.11 Limitations of study

1. In my study, I focus on the input of manipulation and the generation and presentation of forces and torques; I assume the associated visual display exists and performs satisfactorily.

2. Although the ARM has limitations imposed by its mass, backlash, and friction, I do not undertake to improve the mechanical structure.

3. In molecular mechanics, there is no unique way of assigning partial atomic charges to atoms in molecules. I am using the most popular model, that of Kollman, and keep the partial charge mapping in a lookup table [Weiner and Kollman 84]. Other models can be readily substituted.

Chapter 2

A Survey of Force Display and Docking Tools

2.1 Creating an illusion of feel: force display systems

The essence of a virtual world is to simulate the existence of matter by a computer, as Sutherland articulated in 1965 [Sutherland 65]. Sutherland even suggested a sense of feel as a goal in a virtual world, so that the virtual object feels real.

2.1.1 History of teleoperators

Force feedback had been applied to both the physical world and to virtual worlds. In applications to space, undersea and radioactive environments, etc., a user is in control of a master manipulator while the slave manipulator is doing the work remotely. About 1945 the first modern master-slave teleoperators were developed by R. Goertz at Argonne National Laboratory. These were mechanical pantograph mechanisms by which radioactive materials in a "hot cell" could be manipulated by an operator outside the cell. Electrical servomechanisms replaced the direct mechanical tape and cable linkages in 1954 in systems requiring separation of the master and slave [Goertz 54, Goertz 61]. Bejczy designed systems whose slave geometry can be totally independent of that of the master's, as long as motions and forces are faithfully reflected [Bejczy 80]. So, in Bejczy's system, the master manipulator can be a force-reflecting hand controller, while the slave manipulator can be a robot with force sensors.

When the force-feedback idea is applied to the virtual world, the master manipulator becomes a force-reflecting controller (force display) with a computer model assuming the role of the slave manipulator and its task-world. Compared to robotics, there are few results [Bejczy 76, Bejczy 80, Hayward 88] available in teleoperators, and even fewer in virtual-world

force display. Advances in teleoperators help greatly the engineering of a virtual-world force display.

2.1.2 History of force display at UNC

Batter and Brooks began exploring the world of feel at the University of North Carolina in 1972 [Batter 72, Brooks 77]. Batter used a computer to control a two-dimensional manipulator that could simulate various force fields such as gravity, spring, electric dipole, and higher order fields including atomic van der Waals field. The system was tested as a teaching aid for introductory force fields in a physics laboratory course.

Capowski adapted a 6-degrees-of-freedom master-slave manipulator designed at the Argonne National Laboratory (Argonne Remote Manipulator: ARM) for force display; Kilpatrick used it to simulate seven child's blocks on a table [Capowski 71, Kilpatrick 76]. The table and the seven blocks were simulated visually by computer graphics. From this virtual world simulation, Kilpatrick made the following observations:

1. The supplement of force cues to visual cues is a more powerful world-model illusion than stereo images applied to the same visual cues.
2. It is difficult to simulate a hard surface, such as a table-top. The table-top either feels spongy when the hand can stably push against it or becomes unstable during contact when the system has the spring constant stiff enough. Kilpatrick observed that when the sampling frequency was above 25 Hz, the hard surface felt good.
3. The addition of sound feedback (a click sound) during the surface contact is useful. Even though the table-top still felt spongy, the click sound made the subjects believe it to be a realistic table-top. Kilpatrick's conclusion was that multiple-sensory inputs can reinforce each other in creating illusions.

Brooks believed in 1976 that the force-feedback ARM could be useful in simulating the binding forces between two molecules. If chemists can feel the binding forces between two molecules, it would help docking. However, the computer power needed for such real-time force field simulation was estimated to be 100 times more than that available on the then state-of-the-art minicomputers.

2.1.3 Recent force display systems

Faster computers and color graphics engines now enable people to simulate complex force fields such as those in molecules [Ouh-young 88]. Smith used brake motors in creating

realistic spring simulations and fairly good hard surfaces [Smith 88]. At the same time, light-weight and high-torque motors were used to construct a delicate force-feedback joystick that can simulate surface textures [Behensky and Minsky 89].

2.2 Computer simulation of forces

How to create the illusion that one is holding a real object? Let us be more specific. Given a computer-controlled joystick, can we simulate the dynamics of a spring-mass system including its mechanical impedance (mass, stiffness, viscosity)? If we can simulate mass, stiffness, and viscosity, then one cannot tell whether an object is real (using tests of Newton's three laws of motion). In theory we can do this simulation if (1) we can measure the position, velocity, and acceleration of the hand-controller precisely, and (2) can calculate and deliver the outputs continuously and without lag.

Consider the task of simulating a simple spring-mass system. Suppose the simulated mass is at the hand-controller position. In order to know the position and velocity of a mass in motion, the hand-controller trajectory must be known. The input signals from the position, velocity, and acceleration instruments, if analog in form, must be converted into digital form for a digital computer. Similarly, digital output from a computer must be converted into analog signals in order to drive traditional servo motors, otherwise step motors should be used. The computer samples external data at a certain sampling rate, does the force-field calculation, and sends out control signals to servo motors.

2.2.1 What destroys an illusion of feel?

In general, data conversion and computer speed limit the attainable sampling rate. For real-time computer graphics, 15-30 frames/second performance proves enough for acceptable illusions. What is the minimum sampling rate for good perception of force through a human arm? This question relates to human response time, human arm dynamics, system performance, and what is being simulated.

One thing is certain: If the system is inherently unstable, the illusion of dealing with a real object is destroyed immediately. Another criterion common both to computer-generated images and computer-synthesized force fields is that the displayed object cannot jitter if it is supposed to be stationary in time. Noise (quantization noise, thermal noise in potentiometers, noise in transmission lines, and noise in electronic components) in input data causes jitter problems. Bad force-field simulation/system dynamics causes inherent instability, which is intimately related to the sampling period.

2.2.2 An example of instability due to sampling

Consider the simulation of a mass attached to a damped spring (Figure 2.1), we want to show that the system can be unstable due to sampling in the simulation. Ideally when the mass is pulled away and released, the mass will oscillate and finally stop. When sampling period Td is big enough, the system oscillates more and more wildly. Suppose the sampled mass position at time t is in the point A. After Td, the system reacts by pushing the mass to the right, since the mass was reported to be in the point A position. However, the mass center has already moved to point B after Td seconds. Therefore, the feedback force is really pushing the mass farther to the right, thus creating a bigger and bigger oscillation.

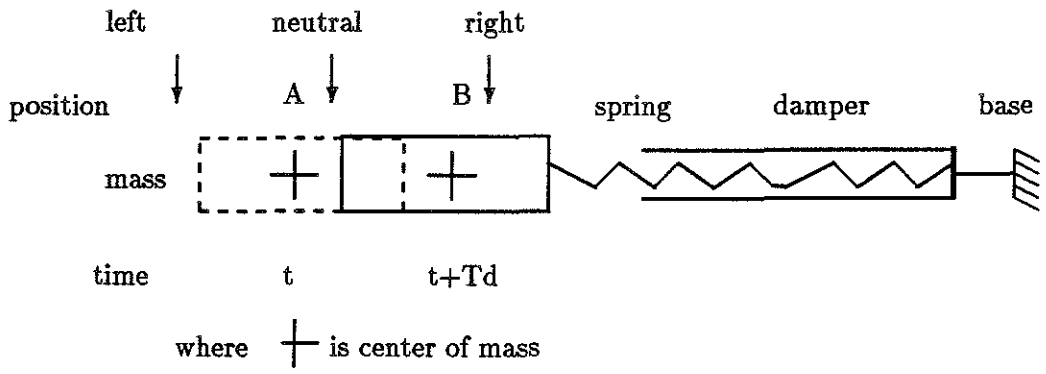


Figure 2.1: A spring and mass system in oscillation

2.2.3 Problems of a man-in-the-loop design

When a user is involved in the control loop, the problem is complicated— the system is far from linear. The human operator's own physical characteristics are involved in the feedback loop, and he changes those parameters dynamically and radically.

Kilpatrick observed that when the sampling frequency was above 25 Hz, the hard surface (table-top) felt good [Kilpatrick 76]. If this 25 Hz is a threshold for the ARM, why? The following question has not been answered analytically, and is crucial to the system design. What is the required sampling rate and latency for the system to be useful for molecular docking?

The answer to this question can best be demonstrated in Figure 2.2, and depends on at least five factors:

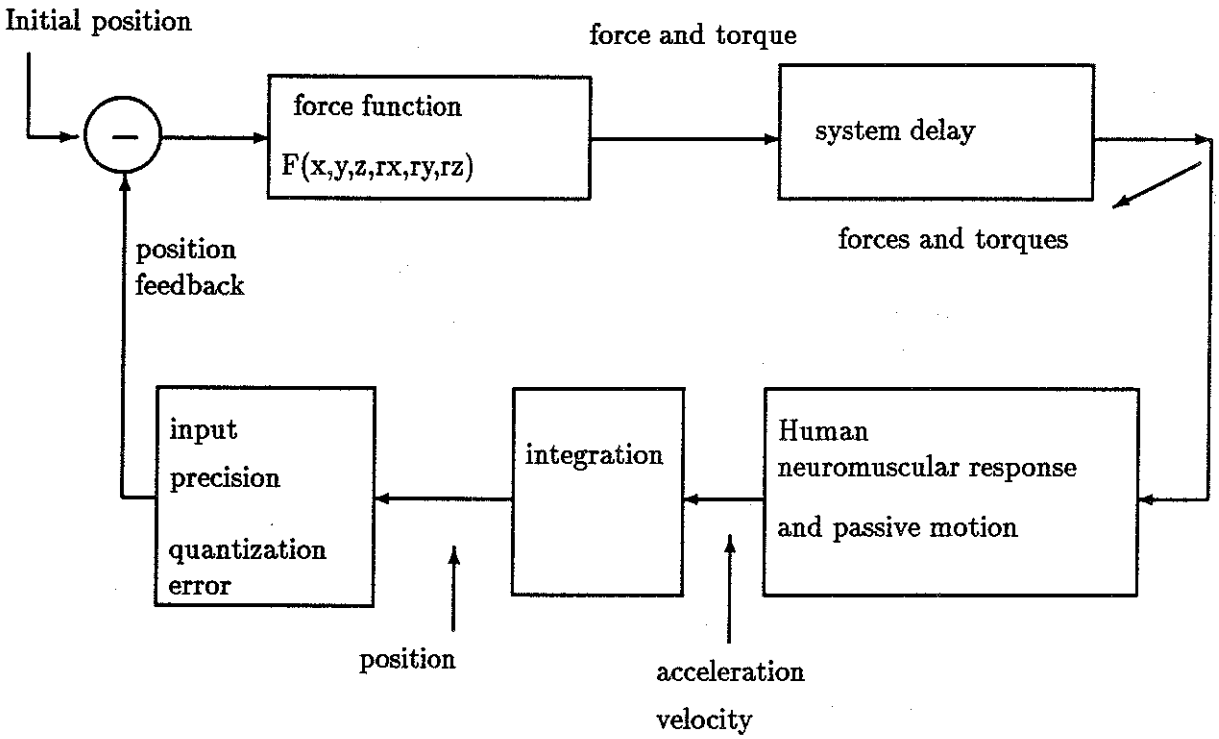


Figure 2.2: A model of the man-in-the-loop force-feedback system

1. The system latency from the user's hand input motion to corresponding force output back to the user's hand.

2. The human neuromuscular response. If it is conscious motion, the latency is around 200 ms; if it is subconscious, the response time can be less [Stark 88].

3. The force field as a function of 6-D positions. Our experience shows that a force function with steep slope will cause the system that has a latency to oscillate longer before coming to its final state, or even to become unstable.

4. Input precision of the ARM. ARM cable stretching, backlash, and quantization error contribute to the total error in input.

5. Involuntary (passive) motion, the passive hand response to a pulse of force from the ARM. This is new to traditional manual-input methods, since this means that the user

is not in total control of the input device.

In chapter 5, the man-in-the-loop problem is addressed through analysis in automatic control theory.

2.3 A survey of docking tools

Biochemists have used computers to aid the design of drugs, and they have developed strategies. Interesting surveys [Kuntz 82, Beddel 84, Janssen 85] on strategies in pharmaceutical research indicate that the future of drug design appears promising.

In order to model molecules, one needs models of molecular force fields. Quantum mechanics is a theoretical model which describes the molecular orbital functions (by solving the Schrödinger equation), but its complexity of computation limits its application to small molecules only. For macromolecules, empirical models are used. A complicated model [Dean 87] usually has terms of

1. Van der Waals and short-range (nuclear) forces
2. Electrostatic forces and hydrogen-bond forces
3. Molecular dynamics (including additional terms of bond stretching, bond angle bending, and torsion angle twisting)
4. Hydrophobic forces (in aqueous solvent).

Based on the above model, we classify the molecular docking problem into several levels of sophistication. (1) Geometry docking, where both the drug and receptor molecule are rigid bodies with fixed conformation. (2) Docking with deformable drug molecules. The drug can have bond twisting, for instance. (3) Docking with deformable drug and receptor molecules. For example, the receptor molecule can have side-chain twisting. (4) Docking with full molecular dynamics. This may include interactions of aqueous solvent. Our current docking system belongs to level (2) above, mainly because of its low computation complexity.

Once a molecular force field is given, minimization of its energy can predict the molecular structure. There are two popular programs that are used as molecular force-field minimizers— AMBER for molecular mechanics and GAUSSIAN for quantum mechanics [GAUSSIAN 85].

1. AMBER: This is a gradient-based energy minimizer, and is efficient in local energy-minimization. It takes a few minutes to hours of computation for each convergence, and therefore is suitable for batch mode computation.

2. GAUSSIAN: It approximates the molecular orbital functions as linear combination of Gaussian functions, and needs hours of computation on a supercomputer even for small molecules.

Usually, for a particular local energy minimum, the above two approaches will give a reliable solution. However, a good initial guess (at the neighborhood of the global minimum) should be given in order to find a global energy minimum, otherwise the solution is a local minimum. Therefore, the above two methods are usually combined with other interactive tools.

Systematic search of the configuration space is extremely costly. Even if the molecule is treated as a rigid body with rotatable bonds, a brute-force systematic search has to deal with the following parameters:

1. 3 D.o.F in translation, 3 D.o.F in rotation. This is the motion of the drug molecule relative to the receptor molecule.

2. N D.o.F of rotatable bonds in the drug.

3. M D.o.F of rotatable bonds in the receptor. In general, $M + N$ can be a big number for a drug-receptor complex.

The time complexity of brute-force systematic-search would be $O(x^{(M+N+6)})$, where x is the time for 1-D search.

Considering the exponential complexity, brute-force systematic-search is out of question, except for trivial cases involving two or three rotatable bonds in state-of-the-art main-frame computers. The following algorithms use smart strategies in their searching.

2.3.1 Docking by global optimization algorithms

Lock and key method. The receptor and ligand are considered as rigid bodies composed of atomic spheres which define respectively a "lock" and "key" [Kuntz 82]. Once the two structures are defined, a geometrical match between the key and all possible key-holes has to be searched for. For example, the myoglobin-haeme binding generated 573 feasible arrangements. When docking is only partial and much of the ligand surface in the bound state is accessible to solvent, this algorithm would not be efficient.

The Ellipsoid algorithm. The Ellipsoid algorithm is used to do optimization based on a set of inequality equations [Ecker 83, Billeter 87, Billeter 88]. The original search space is represented as an ellipsoid, and after each iteration, the volume of the new ellipsoid is reduced. The final ellipsoid, if every inequality equation is satisfied, is the solution space.

When the Ellipsoid algorithm is applied to molecular docking, first the constraints are represented as inequality equations. For example, the constraints include the upper-limit distance for two atoms that form a hydrogen-bond and the lower-limit distance for two nonbonded atoms. The results were promising: 80% of the solutions did not violate any constraints, and some of them resembled the crystal structure.

The advantage of this method is that it combines knowledge (constraints) in its implementation. The limitations are that (1) only low dimensions (up to 50) can be handled efficiently, otherwise the convergence is very slow, and (2) there is no guaranteed convergence.

Molecular dynamics and simulated annealing. This belongs to probabilistic/statistical optimization and can be applied to docking problems in which both the receptor and the ligand are flexible in structure [Brunger 88]. The principle is to heat up the whole system and let it cool down slowly. The final cooled-down structure is the solution. The problem is that the simulation time is extremely long, even for supercomputers, otherwise it is easily trapped in local energy minima.

Dixon studied nonlinear optimizations and concluded that the mathematical nature of the global-optimization problem is totally different from that of local optimization [Dixon 80]. For all practically useful deterministic methods, all we usually conclude is that while the method may perform well on a set of test functions, it cannot be guaranteed to find the global minimum in every case. When it does, it usually takes a long time for high D.o.F. systems.

2.3.2 Docking by man-machine interaction

The following systems include user manipulation in seeking docking positions. I list them in chronological order of publication.

Pattabiraman's tool [Pattabiraman *et al* 85]. Here the main feature is real-time energy calculation by 3-D tabulation for nonbonded interactions, while full-scale calculation takes minutes for one iteration. The Simplex minimizer is used to minimize the energy. The

interaction of thyroid hormones and its analogues with prealbumin was studied, and the results qualitatively predict the order of preference of binding as observed by experiments.

O'Donnell's tool [O'Donnell 87]. Here the motion of molecules is controlled by an electromagnetic control wand (a 3Space tracker, similar to the Polhemus), which is the main feature of the system. In one experiment, the subjects tried to superimpose the previously docked tripeptide (as a reference). Subjects did the task within 15 seconds, and the average RMS difference in location between the two tripeptides was 0.012 nm. Stick models are displayed on a vector display device with stereo viewing. A function key starts an energy minimizer interactively.

Palmer's Docktool [Palmer 87]. This system was implemented in a raster graphics system (Ikonas). The biochemist docks a movable stick-figure drug molecule into a static protein represented by double-sized spheres. The surface defined by the surfaces forms a very thin cavity. The biochemist fits the very thin stick figure into the cavity with the same constraints as if both the drug and the protein were represented by proper-sized spheres. Bump checking was easily done (with the technique invented by David Barry of Washington University).

Cambillau's TOM [Cambillau 87]. This is a docking subpackage built into FRODO [FRODO]. The potential energy function includes Coulomb and van der Waals interactions. Interactive energy minimization is done by the conjugate gradient procedure, treating both ligand and parts of the receptor as flexible units. Potential energy function includes terms for nonbonded and torsional interactions. Docking results from a complex between alcohol dehydrogenase (LADH) and a pyrazole derivative have been compared with crystallagraphic results for this complex. The kinetic results and model-building results agree qualitatively.

Faster molecular mechanics calculation. There are ways to speed up molecular mechanics calculations. For example, Karfunkel's algorithm can deal with both deformed drugs and proteins, and its simplified calculation (with elaborated tabulations) still maintains a high precision as compared to the most sophisticated molecular mechanics energy function [Karfunkel 86]. For instance, he defines the interfragmental potential between two fragments (of a protein) separated by one torsional axis as "nearest-neighbor potential". Similarly, "next-neighbor potential" is associated with two torsional axes between two fragments. For practical purposes, about 100 different functions (with one or two torsional angles as parameters) are sufficient for describing all nearest- and next-neighbor interactions in any protein composed of naturally occurring amino acids. This takes care of nonbonded interactions within a protein, and is the highest number of required computations for a flexible protein.

Using a short cutoff distance (5 Angstroms) between nonbonded pairs helps limit the number of possible interactions. The Fast Multipole Method [Greengard 88] uses tree structures to partition space, then uses analytic observations concerning multipole and Taylor expansions to produce results that are accurate to within round-off error. This appears to be a good method for nonbonded interactions.

Karfunkel's assertion is that in order to perform an interactive docking with flexible molecules at reasonable costs, the computing time must be reduced by a factor of 1000 (i.e., from an hour to seconds in the case of a protein consisting of about 200 amino acids). No wonder that from the surveys of [Beddel 84, Janssen 85, Karfunkel 86], a consensus emerges that a docking tool must be manipulated interactively using computer graphics.

There are tradeoffs in all the above systems. Models of rigid structure can be calculated in real-time, models of partly deformable structure can be calculated interactively ("while-you-wait"), and models of fully flexible structure can only be done in batch computation. None of them is so good that it can fully replace others. The best combination of them would be to

1. Use models of rigid structure in real-time for most of the binding to find a crude initial guess.
2. Use models of partly deformable structure interactively— for example, let the user twist the rotatable bonds. Use a gradient-based energy minimizer when a new conformation looks promising.
3. Use models of fully flexible structure in batch mode selectively in order to to study the dynamic behavior.

Chapter 3

A Scenario of Docking Molecules

3.1 Introduction

This chapter describes how the GROPE-III system is used in practice. The ARM system is combined with two graphics systems, Pixel-Planes4 (raster display) and the E&S Picture System PS300 (vector display). The real-time collision detection (bump checking), energy, force, and torque calculations necessary for molecular docking are implemented on a SUN4 workstation. This system is capable of doing more than 30 updates/second for typical proteins and inhibitors.

3.1.1 Docking on E&S PS300

We have used in GROPE-III different graphical models for a variety of molecules. The two most useful models are (1) the stick model (of bonds) and (2) the solvent-accessible surface model with color maps showing the electric potentials (Figure 3.1)[Connolly 83]. The stick model (bonds) can show side chains (easily identified by a chemist) and types of atoms, so that chemists can predict their chemical functions. However, the shape of the active site and the distribution of electric potentials can best be shown by solvent-accessible surfaces instead of by a stick model.

First, a chemist may use the solvent-accessible surfaces to find the binding of two molecules by matching their geometric shapes (mating surfaces) as well as their colors. For example, a concave surface with positive electric-potential distribution (colored blue) is matched to a conjugate convex surface with negative electric-potential distribution (colored red). In the cases of DHFR + trimethoprim and DHFR + methotrexate binding, the above matching is very effective. It is still a very crude binding in molecular docking, but it reduces the otherwise tremendous search space to a few finite regions.

Next, a chemist may further probe each of the possible combination by using the stick model, since he knows the side chains and their chemical functions. The solvent-accessible surfaces usually do not carry information about atom types and the side chains. If all this information is superimposed, the result is often a confusing clutter. Interaction energies between binding molecules are shown as two vertical vectors with their height proportional to their energies (Figure 3.1).

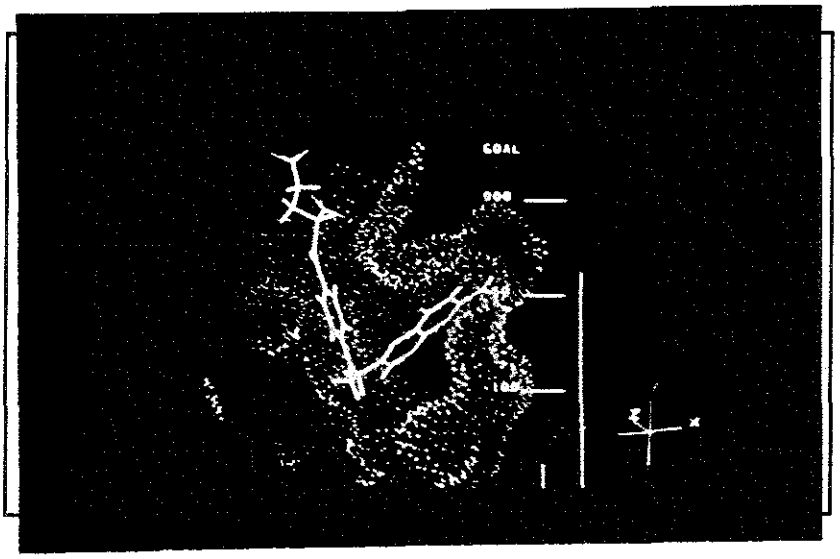


Figure 3.1: A photograph of the PS300 screen with the Connolly surfaces, a stick model, and two energy thermometers.

Real-time bump checking (collision detection) is done using the grid method, which is described in a separate section. A flashing yellow vector is shown between two colliding atoms (in the stick model) or between a colliding atom and a dotted solvent-accessible surface. There is an optional visual display of the total forces and torques exerted on a drug molecule (chapter 4.2). Force feedback through the ARM is combined with all the visual cues. We also provide a run-time energy minimizer (steepest-descent method) that will converge to a local energy minimum from the current location. The chemist therefore uses manual manipulation to find the neighborhood of the global minimum, then uses the algorithm to find the exact minimum.

3.1.2 Docking on Pixel-Planes4

I implemented a demo docking program on Pixel-Planes4 based on the "pphront" interface (using PPHIGS library). Communication between the Pixel-Planes4 host and the ARM host

(a SUN 4) was done by Unix sockets using Ethernet. We soon discovered that we needed stereo and transparency for effective visual representations.

The Pixel-Planes4 version is implemented by taking advantage of raster graphics in order to have a better visualization.

(i) For the solvent-accessible surfaces, display them as either polygons or particles (shaded pixels, implemented by Matt Fitzgibbon) as dotted surfaces. Currently, there are no true transparency functions in Pixel-Planes4 PPHIGS library, so I do not implement transparency for polygons. On Pixel-Planes5, there will be transparency for polygons.

(ii) For the equivalent stick model, display it either as sticks or spheres (space-filling model).

In all the representations, great care must be taken to show the active site of a protein (sometimes a cavity) and the drug molecule such that the drug is not obscured by the protein. If both protein and drug molecule are represented as spheres with their van der Waals radii, some drug atoms are bound to be obscured by the protein. So I do not consider spheres a useful representation, even if we used reduced van der Waals radii to allow more room between the drug and the receptor.

3.2 Bump checking (collision detection)

The active site of DHFR contains approximately 600 atoms; a drug molecule such as trimethoprim contains 40 atoms. For brute-force bump checking, there are 24,000 distance calculations. The above computation would take a Masscomp workstation plus an array processor 0.7 seconds, and would take a Sun4 work station 0.5 seconds. This is far from real-time bump checking.

We tabulate the forces (a 3-D tabulation) and use linear interpolation to get continuous forces. If there are M atoms in a protein, and N atoms in a drug, the complexity of the force calculation is reduced from $O(M*N)$ to $O(N)$ in the 3-D tabulation. As a byproduct of the force calculation, a simple threshold test on the magnitude of forces yields collision detection. In general, if two non-bonded atoms are within their equivalent van der Waals distance, the atomic forces (van der Waals and short forces) will increase in inverse proportion to the 7th and 13th power of their distance. In effect the forces behave as a step function at the van der Waals boundary. Collision detection implemented in this way is precise enough to 0.1 Angstroms within a 0.5-Angstrom-grid tabulation. This precision proves satisfactory for collision detection in molecular docking.

3.3 Miscellaneous functions of the docking system

This section describes different commands used in the system and corresponding functions.

3.3.1 Control dials and software commands

There are 10 numbered dials (knobs) attached to the forearm and arranged in two columns. Three of the dials are reserved for other functions, and are not used currently. Each dial from 1 to 6 controls a particular rotatable bond in the drug molecule. If the drug molecule contains more than 6 rotatable bonds, the 6 dials can be mapped to any subset of all the possible rotatable bonds.

The 7th dial is actually a switch, that can lock or unlock the drug root group, while keeping the other torsional angles free. One feels the forces from bond rotations; however, the forces only affect one's hand motion, not the drug position. This lock/unlock function was suggested by Dr. Lee Kuyper from Burroughs-Wellcome. In my implementation, a drug molecule can be divided into subgroups, so that each subgroup does not contain any rotatable bond within it. Two neighboring subgroups are connected by a rotatable bond. Each subgroup can be activated or deactivated through keyboard functions. Activation/deactivation means turning ON/OFF the force-field simulation for that subgroup, although visually the subgroup is still there. For example, pressing key **B** activates subgroup B.

PS300 dial box and function keys. There is another PS300 dial-box that has eight dials, and the dial functions are tabulated below.

dial labels	function
Disparity	disparity of stereo images
Back/Front	clipping planes
Scale	image scaling
TRANS-X	translate energy thermometers
TRANS-Y	translate energy thermometers
Rot-X	rotate X
Rot-Y	rotate Y

The PS300 keyboard has special function keys. These function keys are switches that can turn on/off the display of an object on the screen. For example, the following objects are controlled separately: the stick model of a protein enzyme and the solvent-accessible surface of a protein-enzyme. There are similar function keys to control models for drug molecules. At one time, the stick model is useful; at other times, the solvent-accessible surface is useful. So the chemists can choose one or both during run time.

3.3.2 Useful functions and visual display

1. Keys: **h**, **m**. Help: key **h**; menu: key **m**. The menu and help will print all the other functions with their control keys.

2. Keys: **&**, **7**. There are labels showing the residue name and sequence number. This information can be turned on (key: **&**) and off (key: **7**) interactively.

3. There are two energy thermometers: one shows the internal energy of a drug molecule, and the other shows the combined intermolecular energy and internal energy. This was suggested by several visitors using the docking system. The internal energy is biased to be zero initially. Whenever there is a bad contact internally, the internal energy goes high. This will tell the chemist that he should resolve this by manipulating the rotatable bonds.

4. Key: **A**. The energy minimizer (key: **A**) is based on the gradient method, and uses a steepest-descent method. The energy minimizer runs for a fixed 20 iterations and stops. If one wants more iterations, one just restarts it, and it will resume from previous results.

5. Key: **E**. There is a way to show the possible hydrogen bonds formed, and the bond length (key: **E**). The collision vector shows what is unfavorable, but the hydrogen bonds show what is favorable. This is very useful to chemists when they are looking for the binding mode. This feature was suggested by Tom Quinn. Pressing key **E** also shows the binding energy using AMBER force field without the 3-D tabulation. This computation takes 3 seconds.

6. Key: **e**. This command gives the binding energy using 3-D tabulation.

7. Keys: **c**, **f**, **b**, **r**. The system can store the intermediate configurations in the memory, and play them back later. There are four functions associated with the play-back:

Key	function
c	store the checkpoint
f	go forward
b	go backward
r	resume the previous configuration

8. Keys: **Z**, **z**, **i**, **#**, **X**, **x**. The system can record the whole docking process at specified intervals (for example, 1.5 seconds per sample, a compile-time parameter). The recorded data, played back like a movie, is important for the analysis of subjects' performance. The other functions of the movie are:

Key	function
i	start the movie
#	stop the movie
X	increase the movie speed
x	decrease the movie speed

9. Keys: **F**, **S**. The motion of the hand-controller is reflected to the screen motion through some scaling. Although 1-to-1 scaling is natural, scaled-down motion for fine-tuning is also helpful. There are two commands (**F**: scale-up, **S**: scale-down) that can adjust the translation and the rotation scaling.

10. Keys: **R**, **d**, **O**. A random number generator can generate random positions, which are then stored as a set of tabulated random positions for later use.

Key	function
d	move the drug to the known crystal binding position
R	move to a random position and orientation
O	move the drug center to the origin

11. Keys *, **8**, "(", and **9** change the force field. Although Kollman's AMBER force field is used, a user can change the weighting on electrostatic and van der Waals forces, so that he can put different emphases at run time. One can easily verify different hypotheses about the dielectric constants used in the force-field calculation. The keyboard commands are:

Key	function
*	double the dielectric constant
8	decrease the dielectric constant by half
(	double the van der Waals terms
9	decrease the van der Waals terms by half

Chapter 4

A Simple 6-D Docking Experiment

As a pilot study in force display, we first studied a simplified docking problem. The user attempts to find the potential energy minimum in a 6-D space defined by six Hooke's Law springs attached to a manipulable object. (This space has no other local minima.) A pilot controlled experiment with seven subjects, twelve trials each, showed that

1. Performance with a force display is better ($p < 0.01$) than performance with a visual display alone.
2. Subjects are able to find the zero force position more than twice as fast with a force display alone than with a visual display alone.

We also describe a new way of graphically representing the resultants of a set of forces and torques acting on a body. Even though the experiment shows force display to be more effective, it also shows that the simple 6-D docking task can reliably be done with this visual display alone.

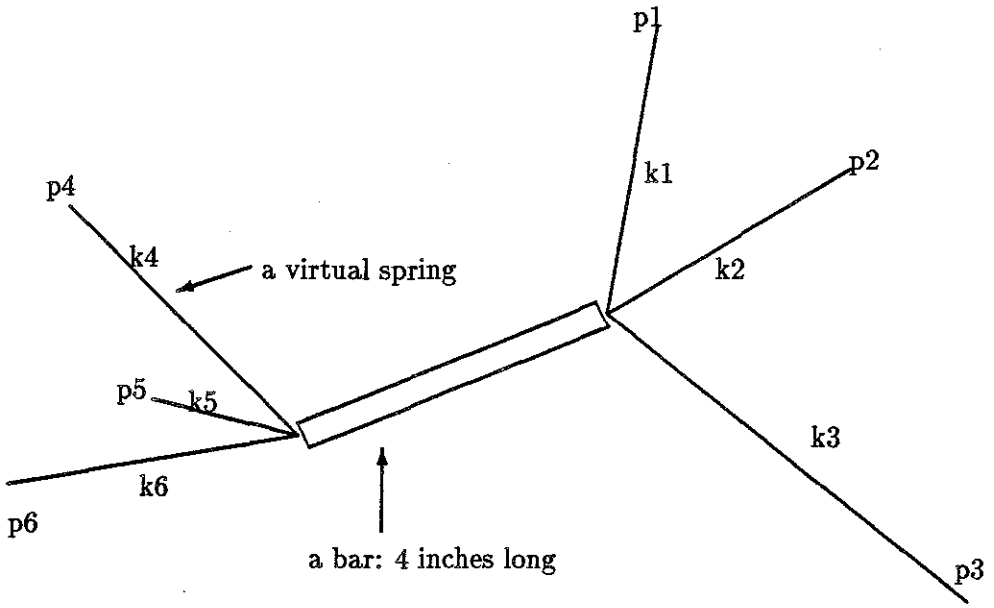
We observe that subjects working with force display move continuously in 6-space to find the minimum, whereas subjects working with visual display alone decompose their activities into 3-D force minimization and 3-D torque minimization. This may account for the two times performance difference. The observed decomposition may be due to the fact that we decompose force and torque in the visual display.

4.1 Introduction

The docking of molecular models requires manipulation in a 6-D force field. We have been studying both visual and force displays to assist chemists in studying such docking. For our pilot study, we needed a simple experiment to see if force feedback really helps. There were several choices for such an experiment, such as "peg-in-a-hole," "block-in-a-cave," and "lock-and-key." However, I needed an experiment requiring energy minimization in a 6-D force field, in order to make it similar to the true molecular force field. I finally chose a six-springs experiment.

4.2 Experimental Set-Up

Simple docking task. The subject is instructed to imagine a 1/2" diameter, 4" long bar, with three springs attached to each end, that he is grasping with a remote-manipulator arm (Figure 4.1). At the beginning of each trial, the springs attached to this imaginary bar are loaded with forces. The subject rotates and translates the bar during a one-minute period or until the spring forces are balanced. A computer-generated beep starts and stops the trial.



The goal is to find the zero-force position and orientation of the bar, where p1 to p6 are positions in 3-D, and k1 to k6 are the elastic constants for springs.

Figure 4.1: The simplified docking task

This simple docking task is a zero-force-finding process. The potential energy of the system is defined as the sum of potential energies of the six springs. When the system's potential energy is minimum, the forces are balanced. For each trial, the system is initially configured so that the normalized system's potential energy is 500 units (an equivalent of about 6 foot-pounds) above the minimum-energy state. We use the system's potential energy as a function of time to evaluate the subject's performance in docking. The task-completion time is the time it takes to get within 2 units of the minimum energy and stay there for one

second.

The virtual world. Manipulation takes place in a virtual world consisting of models of springs. This virtual world is simulated by an interactive computer graphics system plus a force-reflecting controller (a modified Argonne E-3 Remote Manipulator: ARM). The graphics system in our experiment is Evans & Sutherland Picture System PS300, a vector display device. Three-dimensional perception is aided in two ways:

1. Kinetic depth effect is provided by giving the user a dial, held in the free hand, for controlling viewpoint rotation about the Y-axis of the virtual world. The use of this aid costs time. We observed that subjects used the Y-rotation less than 20% of the time.

2. A Tektronix alternating polarization plate gives stereo images. Each subject wore polarized glasses.

On the screen there are two colored spheres landmarking the virtual space. They are unrelated to the docking task, but provide visual cues to the relative positions of objects in the virtual world.

A new visual representation of forces and torques. The resultant of all translational forces on an object is represented as a 3-D vector with one end fixed at the center of the sphere located at the geometric center of the object. The resultant of all torques on an object is represented as a pair of 3-D vectors tangent to the sphere. The length of the force and torque vectors is proportional to the forces and torques applied to the object under manipulation. Visually these vectors would appear as three springs attached to the sphere– one pulling the sphere through the center, the others, tangent, rotating the sphere. There are an infinite number of equivalent torque vectors tangent to a sphere. For any non-zero torque, there are exactly two with origins on the occluding contour of the sphere, except for the degenerate case where all torque vectors are exactly parallel to the viewing plane. We display only the one that is not itself occluded by the sphere. Figure 4.2 shows the visual representation.

In an earlier try, we used a 3-D axial vector at the sphere origin to show the torque, and let the subjects use right hand rule to infer the correct rotation. Subjects did not find this method acceptable. The new visual torque representation has evolved over the past several years. Most subjects quickly understood it, and some volunteered that they found it useful.

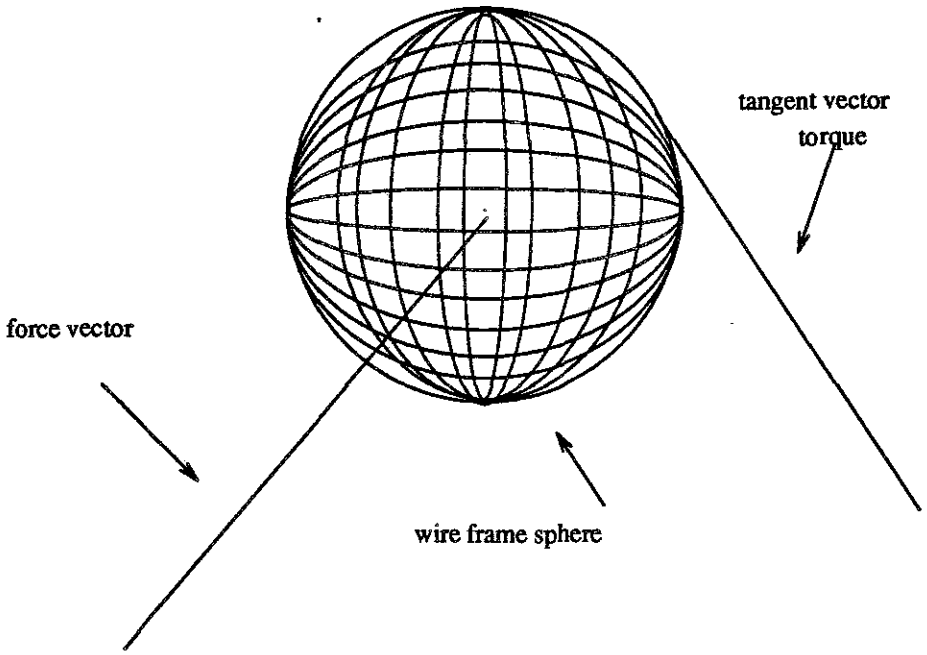


Figure 4.2: Visual representation of forces and torques. Stereo images are presented to the user.

Effects of system latency on performance. Stark found that delays of only 300 msec have clearly measurable effects in one-dimensional visual tracking experiments, and larger delays have greater effects [Stark *et al.* 88]. During the experiment, our 1988 system (Chapter 10.2) had a 12 Hz update rate (The current force system can run at 60 Hz, but the graphics system is limited to about 15 Hz). In our system, the latency between the time the hand controller moves until the time forces are fed back to the hand controller is estimated at two cycles, or 166 msec.

We have done an informal study to determine if lags such as ours give an obvious difference in performance in the case of force fields consisting of springs of relatively small elastic constants. The device for this study was developed by Max Behensky and Doug Milliken, with software by Margaret Minsky, all of MIT. It consists of a 2-D force-reflecting joystick,

and its system latency can be as low as 0.6 ms. Our study showed that when we use small-to-medium range elastic constants for the simulated springs, there are no obvious effects as lag is increased from 2 to 166 msec.

4.3 Experiment

We are interested in the performance of subjects using the following two methods in the simple docking task:

1. Docking with only force feedback, here called **F**
2. Docking with only visual presentation of force and torque vectors. This method is here called **V**.

Our hypothesis was that, compared to **V**, **F** would have a significantly lower mean-energy level after 15, 30, 45, and 60 seconds in a trial.

Procedure. For each trial, the subjects were told to reduce the system energy by minimizing the forces and torques. They were told to do the tests as fast as they could in 60 seconds. After a warning, a beep from the computer terminal signals the start of each trial. Another beep signals a trial's end. Each subject participated in two sessions, each on separate days. Training took about one hour in the first day, and experiment took about one hour in the following day. Each subject was trained with each method until additional training did not improve performance. Training generally took about 20 minutes for **F** and about 40 minutes for **V**.

The computer automatically recorded the subjects' performance in terms of system potential energy every 1/12 second during each 60 second trial. Each trial was followed by a one-minute rest period.

Subjects. We used seven volunteer subjects from graduate students and staff in the computer science department, UNC-Chapel Hill. All subjects were able to see stereo using the polarized glasses. One subject (S2) was very experienced in using the ARM in molecular docking, whereas the other subjects were relatively inexperienced.

Design. The within-subject design is a one-way analysis of variance with repeated measures. Each of the six trials used gave the subject an initial configuration in which the anchor points for the springs were chosen at random. The relative spring constants were chosen at random; the entire set of spring constants was scaled to give an initial energy of 500 units.

The same six trials were used for each method by each subject, but the trials were disguised by changing the initial viewpoint each time the trial was presented. Latin squares were used to permute the methods between and within subjects, to control for learning.

4.4 Results

Table 4.1 shows completion times. Table 4.2 shows the potential energy after 15, 30, 45, and 60 seconds. At time 0, the energy is 500 units in every case. The current system potential energy is a measure of how well the docking has been done so far.

Subject	V	F	mean
S1	57.0	17.7	37.3
S2	18.3	12.4	15.4
S3	45.0	16.8	30.9
S4	37.5	22.4	29.9
S5	29.2	12.9	21.0
S6	53.0	25.1	39.1
S7	50.1	24.6	37.3
mean	41.4 seconds	18.9 seconds	30.1 seconds

Mean completion time in seconds for six trials.

Table 4.1: Performance data for V and F of a simple docking task.

	V				F			
Subject	15 sec.	30 sec.	45 sec.	60 sec.	15 sec.	30 sec.	45 sec.	60 sec.
S1	314.20	141.20	21.45	1.14	3.80	0.52	0.23	0.23
S2	12.56	0.64	0.49	0.49	1.14	0.45	0.45	0.45
S3	152.30	50.10	10.63	1.38	2.99	0.96	0.58	0.53
S4	76.00	12.10	2.63	1.55	9.36	1.49	1.03	1.03
S5	155.70	0.89	0.48	0.45	0.36	0.36	0.36	0.36
S6	232.20	105.50	14.50	1.04	1.72	1.72	0.62	0.62
S7	292.60	131.10	5.79	0.98	5.30	0.30	0.30	0.30
mean	176.5 units	63.0 units	8.0 units	1.0 units	3.52 units	0.83 units	0.51 units	0.50 units

The energy unit used here is scaled to 500 units, an equivalent of 6 foot-pounds.

Table 4.2: Average system potential energy at 15, 30, 45, 60 seconds for six trials.

4.5 Discussion

Energy levels were significantly lower with **F** than with **V** at 15, 30, 45, and 60 seconds (analysis using a Neuman-Keuls posthoc test showed significance, $p < 0.01$, except for 30-second energy levels which showed significance at $p < 0.05$). We observed that most subjects using **F** had reached steady state by 30 seconds into a trial. Some subjects reached steady state using **V** after 30 seconds, but many were still improving their docking position at 60 seconds.

Comparing the mean completion times for the two methods shows

$$\frac{mean(\mathbf{V})}{mean(\mathbf{F})} = 2.2 \tag{4.1}$$

A simple F test on Table 4.1 gives $f(1,6)= 29.7$, $p < 0.01$, and the null hypothesis **V=F** was rejected. A posthoc (Neuman-Keuls) test on our completion time data shows significance ($p < 0.01$) in the pair of means **V,F**.

Almost all subjects said that they are surprised that they can do docking in a **blind** way. They thought that it would take longer to do the job when they were virtually blind. This result may be true only for energy spaces with only one minimum. This may not be the case in molecular docking, where energy minima abound.

The mean final energy for **F** was half as large as that of **V**. The docking trials lasted only 60 seconds, and observation of users informally docking for longer periods causes us to speculate that **V** could produce much better accuracy with longer docking periods.

In fact, it is important to note that **V** did produce a docking solution. Although force and torque assimilation by kinesthesia is more accurate after 60 seconds, a force-reflecting controller is much more costly. There are many applications where force and torque visualization is advantageous, if, by proper visualization, the forces and torques can be understood through vision almost as efficiently as through kinesthesia.

4.5.1 Observations

Axis decomposition of tasks. Our visual presentation of the forces and torques are two independent vectors. During the docking by **V**, we observed that the subjects dealt with force and torque vectors separately. Most subjects shorten the force vector first and then the torque vector. This is not always the best way of minimizing energy, since the orientation

alignment may reduce the system energy more effectively in some of the trials. When the subjects were docking using method **F**, they did translation and rotation at the same time in continuous motion. **F** being twice as fast as **V** also suggests that subjects treated forces and torques as independent entities.

Kilpatrick observed that users of an imperfect-perception visual system, whether interactive computer graphics or closed-circuit television, tended to decompose multi-dimension positioning tasks into several separate subtasks, each of lower dimensionality [Kilpatrick 76]. This is in contrast to normal eye-hand coordination behavior.

A toy problem. Because there is only one energy minimum in the six-springs experiment, any gradient-based algorithm can converge to the solution. We consider this experiment a toy problem, knowing that in real docking problems energy minima abound.

One may ask why not let the ARM handgrip go home alone when there is force feedback? First, one has to press the clutch in the handgrip in order to move the drug molecule. That means one's arm is combined with the ARM system. Second, there is limited ARM working space and the final solution of the suspended bar is beyond the ARM working space. One has to release the clutch and reset the handgrip position several times before he can reach the destination. Third, when the minimum-energy position is within the ARM working space, one can imagine putting a tape on the clutch to active it and let go of the handgrip. The ARM handgrip will reach the minimum-energy position either overdamped (not possible in our system unless a program-controlled viscosity is added) or underdamped. Critical damping is desirable, but not with arbitrary random springs. In the underdamped condition, the oscillation of the handgrip will take even longer time to converge to within a threshold (0.2 % initial energy) of the minimum-energy position. One usually does a better job by using one's arm in reducing the number of oscillations before the final stop.

4.5.2 A hypothesis on task decomposition

Observations on task decompositions lead me to a hypothesis (which others may have stated before): The more axis-decomposition is required in a positioning (orientation) task, the more task-completion time is needed.

It is reported that there would be a speedup of eight times if a 6-D manipulator (Argonne National Laboratory E2) was used (without force feedback) in a positioning task, instead of six separate dials (in rate control mode) for 6-D control [Kim 87].

Our six-springs experiment showed a speedup of two if force feedback ARM was used, instead of visual representation of forces and torques (a decomposition of two).

Therefore, I would assume that a force-feedback system will be useful if it helps the user reduce the number of decompositions in a task. Similarly, a (scientific) visualization method will be useful if it helps the user reduce the number of decompositions in viewing. For example, 3-D volume rendering will be more effective than 2-D contour lines.

If I use the above rules to evaluate the visual force and torque representations used in the experiment, I can say that (i) This new representation has already reduced the six decompositions (if six vectors, three for the force and three for the torque, are used) to two decompositions, (ii) There may exist a new visual representation that can further reduce the number of decompositions to one! However, we do not see one at present.

Chapter 5

A Molecular Docking Experiment

I designed and conducted a molecular docking experiment as my principal study in force display. My hypothesis is that adding force display to an interactive computer graphics system can significantly help in molecular docking problems in terms of task-completion time.

The molecular docking experiment simulated the interaction forces between the dihydrofolate reductase enzyme (over 600 atoms) and six drugs (40 to 60 atoms each). Twelve biochemists tried to dock the drugs into the enzyme.

The experimental results corroborated ($p < 0.05$) my hypothesis in 6-D rigid-body manipulation. There was, however, no significant difference in the case of one-degree-of-freedom chemical-bond rotations. That is, for this 1-D task, the visual cues sufficed. From limited case by case studies, the subjects using the current ARM system are doing faster and getting more precise docking results than those using the Ellipsoid algorithm (one of the best algorithms to date), both in the number of well-formed hydrogen bonds and in displacements.

5.1 Introduction

In chapter four we studied a simple 6-D docking task, but there are more questions to be answered. What would be the performance of force display in real molecular-docking situations? Can the simple docking results be generalized to real applications? What do case by case studies say about how good the experimental results are compared to those done by algorithms only? These are the questions addressed in this experiment.

5.1.1 Hypothesis

I hypothesize that the addition of force display to an interactive computer graphics system can significantly help in molecular docking problems in terms of task completion time, and that molecular scientists can direct the fitting of the drug molecule in a receptor molecule

so that interaction energy is in the neighborhood of the true global energy minimum, given assistance such as visual and force display.

In short, the experiment hypothesis is: task-completion-time (TCT) using force-plus-visual display is less than the task-completion-time by visual display alone.

5.1.2 Apparatus

I used the GROPE-III molecular docking system described in chapters two and three. Two vertical energy thermometers on the PS300 (vector display) screen display the current binding energies. One thermometer shows the intramolecular energy of a drug only, whereas the other shows the sum of intermolecular energy (between the drug and the enzyme) and intramolecular energy. Chemists need to know the sources (intra- or intermolecular energy) of binding energies in order to take different actions to minimize them, so our system provides two independent energy thermometers. If there is no force feedback from the ARM, these constitute the only visual feedbacks that indicate the binding energy.

The protein enzyme and the test drug molecules are displayed all the time in colored stereo images. If a drug-molecule atom bumps into a protein-molecule atom, a golden flashing vector shows the collision. Any collision guarantees a bad binding position. A subject can adjust the magnitude of the forces coming out of the ARM with a dial, and he can turn off the force output by lifting his foot from a foot switch.

5.2 Method

This experiment let biochemists solve real problems in molecular docking. Because of the essential complexity, I have to give up some tightness of control. For example, the test drug molecules, being real molecules and limited in number, may not come from the same sampling distribution, and since the same drug cannot be used by one subject twice, a complete factorial design is impossible.

5.2.1 Subjects

The subjects are twelve experienced biochemists from UNC-CH, Duke University, and Burroughs-Wellcome Research Center at Research Triangle Park. Ten are postdoctoral researchers in chemistry/biochemistry, and all have worked on molecular modeling problems for at least two years.

5.2.2 Procedures

This is a one-way design with repeated measures. The two methods used are

1. Use the two visual energy thermometers and no force feedback from ARM in docking, NF (no force).

2. Use both visual energy thermometers and force feedback in docking, F (with force).

The independent variable was whether or not the subjects were using synthesized force feedback from the ARM. The subjects were to dock the drug so that the binding energy fell within a threshold (10 Kcal/mol) of the known binding energy. The binding energies and inhibitor-enzyme conformation were recorded by the computer and a message "GOAL" was shown on the screen when the energy fell within the threshold.

There was a practice/training section so that the subjects could become familiar with the operations. Two drug molecules were used in the training sessions. Each subject practiced until he reached a criterion (20 minutes to reach goal). Before each subject conducted an experiment, he was required to pass the criterion again just before the experiment began, if previous training was not given on the same day.

The research design is shown in Fig. 5.1. There are four test drug molecules (d1,d2,d3,d4), not used in the training, which can bind to a protein enzyme. I assumed the difficulty of the binding problems for these four test drugs to be about the same. I randomly divided the four test drugs into two groups (group A, group B), each group having two drugs. Docking for drugs in Group A was to be done by F, while docking for group B was done by NF.

There are 6 different ways of grouping, if ordering is ignored. If ordering is considered, there are 24 conditions. Of these 24 combinatorial combinations, I chose 12 and assigned them to the 12 subjects. The 12 combinations were selected based on the rule that if (d_i, d_j) was in group A, and (d_m, d_n) was in group B, then either $(i > j)$ and $(m > n)$ or the inequality sign was reversed simultaneously. The result was that all four drugs had equal probability in the experiment ordering, and equal probability between the two groups.

Each of the 12 subjects was randomly assigned to the 12 conditions. The two methods (F and NF) were used by each subject, and the ordering was counterbalanced by doing either (F,NF,F,NF) or (NF,F,NF,F). The exact ordering is given in Fig. 5.2. The raw data is in Fig. 5.4.

Subject	treatments			
	F		NF	
S1	d1	d2	d3	d4
S2	d2	d1	d4	d3
S3	d1	d3	d2	d4
S4	d3	d1	d4	d2
S5	d1	d4	d2	d3
S6	d4	d1	d3	d2
S7	d2	d3	d1	d4
S8	d3	d2	d4	d1
S9	d2	d4	d1	d3
S10	d4	d2	d3	d1
S11	d3	d4	d1	d2
S12	d4	d3	d2	d1

d1,d2,d3,d4 are test drug molecules
S1 .. S12 are 12 subjects
F: force display + visual display
NF: visual display alone

Figure 5.1: Treatment by subject design.

The subjects were allowed 2.5 hours (more than enough for four test drugs) to dock the four drugs from a random starting position, with five minutes of rest between each docking. The subjects were given three minutes to study the geometries of the next test drug before beginning actual manipulation. The subjects were to dock the drug so that the binding energy fell within a 10 Kcal/mol range of the pre-defined binding energy. The binding energies were obtained from (1) x-ray crystallographic data for the DHFR enzyme with the drug docked (two of the drugs have such data), (2) docking done by a biochemist for two drugs, without crystallographic data bound to DHFR, together with predicted binding energy from laboratory experiment on binding affinity [Kuyper 82]. In general, our test drugs have binding energies ranging from -40 Kcal/mol to -120 Kcal/mol.

For further analysis, the computer recorded the drug conformation (atom translations, rotations, and bond angles) and the binding energy every 1.5 seconds during the docking. This information can be analyzed as a function of time. Therefore, the binding energy and drug conformation are also dependent variables. These variables are interesting to the system designer as a way to observe the behavior of subjects.

Subject	treatment order			
S1	d1 F	d3 NF	d2 F	d4 NF
S2	d4 NF	d2 F	d3 NF	d1 F
S3	d1 F	d2 NF	d3 F	d4 NF
S4	d4 NF	d3 F	d2 NF	d1 F
S5	d1 F	d2 NF	d4 F	d3 NF
S6	d3 NF	d4 F	d2 NF	d1 F
S7	d2 F	d1 NF	d3 F	d4 NF
S8	d4 NF	d3 F	d1 NF	d2 F
S9	d2 F	d1 NF	d4 F	d3 NF
S10	d3 NF	d4 F	d1 NF	d2 F
S11	d3 F	d1 NF	d4 F	d2 NF
S12	d2 NF	d4 F	d1 NF	d3 F

d1,d2,d3,d4 are test drug molecules
S1 .. S12 are 12 subjects
F: force display + visual display
NF: visual display alone

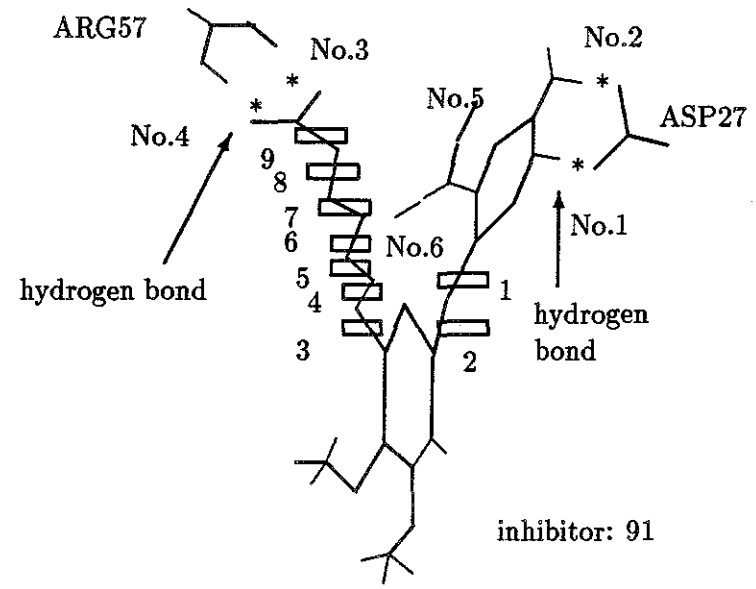
Figure 5.2: Ordering of treatment by subject design.

Test drug molecules. The ARM system can simulate the interaction forces between a dihydrofolate reductase enzyme(DHFR) and six inhibitors (methotrexate, trimethoprim, two carboxy-substituted trimethoprim analogues, one folic acid, one methotrexate analogue) [Baker 81, Kuyper 82]. Trimethoprim (an anti-bacterial drug) and methotrexate (an anti-cancer drug) were used for training only.

Each test drug has from 4 to 12 rotatable bonds, and because of the need of equal difficulty in the molecular docking tasks, I chose the six innermost bonds near the base ring group to be flexible at first, while the others were preset to a good configuration. Later, when the subjects had reached the goal, they were allowed to manipulate additional rotatable bonds.

The starting conformation for each drug molecule was in a non-optimized state (both orientation and translation are randomized), and all the rotatable bonds were randomized before the experiment is conducted.

Useful docking strategies. Through the training session, each subject had developed his own docking strategy. However, the following were common ones, and were highly recom-



means rotatable bond.
ASP27 and ARG57 are residues of DHFR.
* means a well-formed hydrogen bond.

Figure 5.3: Inhibitor 91 bound to DHFR in crystal structure.

mended in oral instruction before the formal experiments.

1. Stage 1: docking by parts. Each drug is subdivided into sub-groups, with one rotatable bond connecting two neighboring sub-groups (Fig. 5.3). There is a major ring (either pyrimidine ring or pteridine ring) in the drug molecules that are essential for the binding. Work on this sub-group first, and ignore the other parts by deactivating them in force simulation. Subsequently, activate the other groups when necessary. Finally, the whole drug molecule is activated. Because this step does not involve any bond rotations, the manipulation is basically a rigid body with 6 degrees of freedom.

2. Stage 2: bond rotations. Once the best fit is found from stage 1, work on the other rotatable bonds. Because of the way in which the rotatable bonds are arranged in our test drugs, most of them are serially linked. The strategy is "backbone to side-chains," that is, manipulating first the bonds that are closest to the base ring. Each rotatable bond has one degree of freedom, associated with one dial. Subjects can manipulate two dials with both hands simultaneously, but in practice they do so only occasionally. This technique requires much experience in hand-eye coordination. Most of the time,

subjects do one bond rotation at a time.

The molecular docking process consists of alternating sessions of stage 1 (6D rigid-body manipulation) and stage 2 (bond rotations). When "GOAL" appears on the screen, it signifies the end of the timed docking task, but the subjects are encouraged to fine-tune the drug positions so that a better fit can be obtained.

Individual docking strategies include

1. When to choose solvent-accessible surfaces or stick models for the enzyme
2. Which view-point to choose and when to change it
3. When trapped in a local energy-minimum position, how to get out of it
4. More thinking and observation and less trial-and-error, or vice versa.

Useful visual feedback To find the best fit is to have an overall minimum binding energy, which implies having formed all possible hydrogen bonds between the enzyme and the drug, and that the drug-enzyme complex is collision free.

In addition to the two energy thermometers, there are two more visual feedbacks. The flashing vectors from bump-checking tell a subject where not to go. This is a negative feedback.

To show the possible hydrogen bonds formed with a vector connecting two atoms is a positive feedback, because it tells a subject where to go. Our implementation does not maintain a dynamic display of the hydrogen bonds, but it shows them upon the subject's request (by a command). Although only one key was needed to enter the command, to reduce the burden of the subject, the author acted as a speech recognizer and typed the keyboard command. The same procedure was followed when any other keyboard commands were required.

The above information provides a visually rich environment. Actually, this environment has many useful dynamic or interactive feedbacks that other existing molecular-modeling tools do not normally provide. Our molecular docking system is an evolving one; each time a pioneer user provided suggestions, we seriously considered them and implemented most of them. All our pioneer users are experts in molecular modeling, and they know what they really need, once they have had a chance to use a modeling system.

We are strong believers of the progressive refinement strategy in designing a new system, and the strategy involves cycles of three phases: implementation, demonstration, and collecting suggestions. The truth is that the biochemists themselves do not know the ideal specifications of a new tool, although conceptually they know what they need. During the demo-implementation cycle, I implemented new functions based on their suggestions, and together we explored the usefulness of each new function.

5.2.3 Results

A brute-force combinatorial search of the conformation space for the best binding energy of this enzyme-drug system takes at least a week on an IBM (3081) mainframe computer for any one of the test drugs [Kuyper 89]. So the probability of guessing a correct solution by chance is extremely small.

	d1: 301			d2: folate			d3: 91			d4: 309		
S1	340	946	F	140	1292	F	600	871	N	156	400	N
S2	643	1220	F	546	878	F	743	867	N	645	887	N
S3	117	772	F	140	928	N	280	525	F	174	379	N
S4	455	735	F	278	538	N	254	408	F	143	1596	N
S5	163	325	F	410	725	N	176	457	N	218	954	F
S6	78	325	F	200	855	N	205	556	N	137	273	F
S7	374	410	N	275	1008	F	179	309	F	325	1430	N
S8	951	1118	N	800	923	F	291	437	F	527	1063	N
S9	296	1019	N	223	418	F	330	611	N	244	668	F
S10	540	631	N	310	1211	F	286	566	N	439	1003	F
S11	499	566	N	533	998	N	244	371	F	117	670	F
S12	234	444	N	182	327	N	200	468	F	187	655	F
	6D	ALL		6D	ALL		6D	ALL		6D	ALL	

All scores are in seconds
d1,d2,d3,d4 are test drug molecules
S1 .. S12 are 12 subjects
6D means 6-D rigid body manipulation
ALL means 6D plus bond rotations
N: without force, F: with force

Figure 5.4: Performance data for F and NF in molecular docking, raw data.

method / subject	F	NF	mean
S1	480 sec.	756 sec.	618 sec.
S2	1189	1388	1288.5
S3	397	314	355.5
S4	709	421	565
S5	381	586	483.5
S6	215	405	620
S7	454	699	576.5
S8	1091	1478	1284.5
S9	467	626	546.5
S10	749	826	786
S11	361	1032	471.5
S12	387	416	401.5
mean	573 sec.	745.6 sec.	659 sec.

source	df	SS	variance	F
Treatments	1	178,020	178,020	6.32
Subjects	11	2,292,871	208,442	
T x S	11	309,890	28,172	
Total	22	2,780,781		
$p < 0.05$				

where SS is sum of squared deviations, T x S is treatments-by-subjects interaction, df is degree-of-freedom.

Figure 5.5: Performance data for F and NF in molecular docking, 6-D rigid-body manipulation.

The results are tabulated in Figures 5.5, 5.8, and 5.9. The analysis of variance method (ANOVA F-test) and t-test are used. My pressumption is that the four test drug molecules are similarly difficult to dock, so that the difference among drugs will not greatly bias the performance data. I have taken three steps:

1. Although some drug molecule may have more than ten rotatable bonds, I limited three test drugs (d2,d3,d4) to have six rotatable bonds each, and the fourth drug (d1) to have only four rotatable bonds, since d1 has only four.

2. To make test drug d1 appear to be similarly difficult, the energy threshold was made smaller (8 Kcal/mol, instead of 10 Kcal/mol).

3. I counterbalanced the drugs in the experiment.

The following data show the variance of docking-difficulty for test drugs. Fig. 5.6 is derived from Fig. 5.4, and lists all data with NF method under each test drug. From Fig. 5.6, if we use t-test for pair-wise comparisons among drugs, the biggest t-test value among all pair-wise comparisons is 1.43, which is smaller than the critical value of 2.228 (t distribution for $\alpha = 0.05$). There is no significant difference for rigid-body 6-D manipulation of the test drugs, although the sample size (six samples for each drug) is small. Similarly, in 6-D rigid-body manipulation plus bond rotations (see Fig. 5.7), the biggest t-test value between the two drugs (309 vs. 91) is 1.27. Again, the null hypothesis cannot be rejected.

Fig. 5.5 shows the times used in 6-D rigid-body manipulation (by summing periods from the onset of the 6-D manipulation to the onset of bond rotations). Each subject used four drugs, two under F, two under NF. The score under F for each subject is the sum of the two task-completion-times for the two drugs under F. The NF scores are similarly arrived at. An ANOVA F-test on Fig. 5.5 gives $f(1,11) = 6.72, p < 0.05$, and the null hypothesis, F equals NF, was rejected. The t-test value is -2.51, the critical value is -2.2, so we reach the same conclusion. The ratio NF_{mean}/F_{mean} is $745.6/573 = 1.30$, meaning that the speedup by F over NF is about 30%. The difference between NF and F is $745.6 - 573 = 173$ seconds.

Fig. 5.8 contains the time used in bond rotations (by summing the periods from the onset of bond rotations to 6-D manipulation). This figure is derived from Fig. 5.4 by subtracting the 6-D column from the ALL (6-D + bond rotation) column. An ANOVA F-test on Fig. 5.8 gives f(1,11) = 0.004, and the null hypothesis (F equals NF) cannot be rejected. Similarly, the t-test value is 0.07, far smaller than the critical value ($\alpha = 0.5$) of 2.2. Fig. 5.9 shows the task-completion times for the whole docking period (6-D rigid-body manipulation plus bond

301	309	91	folate
374 sec.	156 sec.	600 sec.	140 sec.
951	645	743	278
296	174	176	410
540	143	205	200
499	325	330	533
234	527	286	182
482 sec. mean	328 sec.	390 sec.	291 sec.

T-test: 301 vs. folate, t = 1.43, and 301 vs. folate, t = 0.60

Figure 5.6: Performance data for four test drugs with NF method in 6-D rigid-body manipulation.

301	309	91	folate
410 sec.	400 sec.	871 sec.	928 sec.
1118	887	867	538
1019	379	457	725
631	1596	556	855
566	1430	611	998
444	1063	566	327
629.7 sec. mean	959.2 sec.	654.7 sec.	728.5 sec.

T-test: 309 vs. 91, t = 1.27, and 301 vs. 309, t = -0.99.

Figure 5.7: Performance data for four test drugs with NF method in 6-D rigid-body manipulation plus chemical bond rotations.

method / subject	F	NF	mean
S1	1785 sec.	515 sec.	1136.5 sec.
S2	909	366	637.5
S3	900	993	946.5
S4	434	1713	1073.5
S5	898	596	747
S6	383	1006	694.5
S7	863	1141	1002
S8	269	703	486
S9	619	1004	811.5
S10	1465	371	918
S11	680	982	831
S12	736	355	545.5
mean	826	812	819

Source	df	SS	variance	F
Treatments	1	1,190	1,190	0.005
Subjects	11	929,281	84,480	
T x S	11	2,904,203	264,018	
Total	22	3,834,674		
$p > 0.05$				

where df is degree-of-freedom, SS is sum of deviations,
TxS is treatments-by-subjects interaction.

Figure 5.8: Performance data for F and NF in molecular docking, flexible chemical-bond rotations.

rotation). An ANOVA F-test on Fig. 5.9 gives f(1,11) = 0.37, and the null hypothesis cannot be rejected. Similarly the t-test value is -0.61, greater than the critical value ($\alpha = 0.05$) of -2.2. The difference between NF and F is 1520 – 1399 = 121 seconds, or 13%.

5.2.4 Practice effects

The practice effects are plotted in Fig. 5.10. The overall docking times show a positive-practice effect from the first trial to the second trial, and a negative-practice effect from the second to the fourth trial. There is a similar trend in 1-D chemical-bond rotations. The positive-practice effects could be caused by experience; the negative-practice effects could be from the fatigue effects. The first two trials could be exciting to the subjects, but by the time the forth trial was given, all subjects had spent more than one hour in intensive docking and could be fatigued.

In the 6-D rigid-body manipulation, there is a negative-practice effect from the first trial to the second one, and there were no practice effects in the later trials.

5.2.5 Discussion

In Fig. 5.5 the effect (task completion time) between F and NF is significantly different ($p < 0.05$), and F is better than NF. This result supports my hypothesis: docking rigid molecules in 6-D with visual feedback alone is more difficult. Force display can be a useful adjunct in molecular docking.

Our system does not provide direct force feedback in dials, so the internal force within the drug cannot be reflected. However, the drug interacts with the enzyme, and the conformational change in the drug is reflected **indirectly** at the hand-control from the interaction force. There are times when the bond rotation drastically changes the intra-molecular forces, but at the same time the inter-molecular forces are relatively small, so even the indirect force is missing. This means that there is no force feedback in this condition.

In Fig. 5.8 there is no significant difference between F and NF, when bond-rotation results are considered. The results show that visual display was rich enough, and the addition of force display (indirectly) did not help the docking. This is reasonable. After all, the force display for bond rotation is 1-D in nature, and the visual energy thermometer is a very good 1-D feedback. At the same time, the flashing bump vector connecting two atoms in collision provides 3-D information as to where to avoid collision.

Fig. 5.9 contains the entire task-completion times for a test drug, including 6-D rigid-body manipulation and single-bond rotation. There is no significant difference between F and NF. This needs more explanation. First, the difference between two means of F and NF in Fig. 5.5 is 173, and the same difference in Fig. 5.9 is 121. It appears that most of the effects of F can be attributed to Fig. 5.5. In Fig. 5.8, there is essentially no difference even in the mean task-completion times. The mean time spent in 6-D rigid body manipulation is 745.6 sec. by NF, and 573 sec. by F, and the mean time spent in bond-rotation is 812 sec. by NF, and 826 sec by F. The ratio between the two different kinds of manipulation (6-D rigid body manipulation over 1-D bond-rotation) is approximately 0.9. That is to say, a subject spent about equal time in each operation. The variance in the total task-completion time is so big when 6-D rigid-body manipulation plus single-bond rotation are considered that the final statistics using twelve subjects do not show any significant difference between F and NF.

5.2.6 Path of molecular docking

It seems that the force display contributes to a 30% speedup in the 6-D rigid body manipulation. But why 30% only, instead of the 2 times difference described in the simple 6-D docking task (six springs) before? The path (trajectory) of the drug molecule provides some explanations.

I define the path as the trajectory integrated over time. The rotation in a short time interval can be linearized and expressed as a space angle around a fixed 3-D vector. In this way, both translations and rotations in the molecular docking can be measured precisely. The path length is given in Figure 5.11. In order to have rotations converted into Angstroms in displacement, I let a drug molecule be simplified and represented as a rod of ten Angstroms in length (with all atoms evenly distributed within the rod); so the average displacement of the rod with 1 radian rotation is 2.5 Angstroms.

Study of the paths reveals that

1. The path of the drug molecule showed that the average path lengths were more limited in F than in that of NF. The ratio $NF_{meanpath}/F_{meanpath}$ is 232.4/164.6 = 1.41; the path length of F is 41% shorter.

2. When the drug was heavily entangled with the enzyme, subjects tended not to use force feedback (by lifting the foot on the foot switch). The turbulent force generated by many atoms colliding simultaneously is counter-productive if the subject really uses it. The time involved when there are more than two colliding atoms is approximately 30% of the manipulation time, and I assume the subjects did not use force feedback during this

period. If this 30% time (without using force) is accounted for, the speedup using F is estimated to be 50%. The reason is that the true manipulation time done with force feedback is actually 500 seconds, instead of 573 seconds; if we assume 30% time without force will be $745.6 * 0.3 = 224$, then 100% time with force is $(573 - 224)/0.7 = 500$.

3. The subjects were doing multiple energy-minimum searching. Part of the subjects' time was spent in changing viewpoints, thinking and post-verification (to show the current possible hydrogen bonds visually) to see if the current solution is indeed the global minimum. We can detect the time when there is no 6-D rigid-body manipulation and bond rotation. This time is almost the same in both methods (509 seconds in F, and 513 seconds in NF), independent of 6-D rigid-body or 1-D bond-rotation manipulation, and accounts for 36% (513/1399.5) of the time in F and 34% (509/1520) of the time in NF. This is the part where knowledge in biochemistry plays an important role. In comparison, the previous 6-D docking task of 6 springs involves one energy-minimum only, and no thinking is required.

Assuming that a subject uses exactly the same "thinking time" in F and NF, the actual manipulation times in 6-D rigid-body manipulation are

1. 320 seconds in **F**, 500 seconds in using forces, and 573 seconds in recorded data. The 180 seconds come from the estimated 36% "thinking" time.

2. 565 seconds in **NF**, 745.6 seconds in recorded data, and the same "thinking" time (180 seconds) as above.

3. The speedup becomes $565/320 = 1.77$, i.e., 77% speedup.

In addition to the possible changes in speedup, notice that the more "thinking time" one has in an experiment, the greater it will reduce the power of an experiment by increasing the variance without increasing the difference in mean. This implies that more subjects are required to reach the same (statistical) significance level.

5.3 Observations

This section contains observations and subjects' comments on the molecular docking experiment. While these observations are not verified by rigorous statistics, they sometimes show important ideas.

5.3.1 Results of solving a docking problem using the GROPE-III

Precision of molecular docking. This section gives the docking results in two measurements: (1) hydrogen-bond separations and counts, and (2) displacement (correlation) between a reference molecule (crystallographic data) and experimental results.

Test drugs with known crystal structures

For two of our test drug molecules (91 and 309), crystallographic data for the drug docked are available– the correct docking is known (Fig. 5.3, and Fig. 5.12). These data are used as the reference for our experimental results. Inhibitor 91 shows stronger binding affinity than 309, and both 91 and 309 bind more strongly than does trimethoprim. This is because 91 and 309 have an additional carboxy group that can form hydrogen bonds to the residue Arginine 57. This was exactly the design purpose when these two drugs were made [Kuyper 1980]. Kuyper used a series of carboxy-substituted trimethoprim analogues to test his hypothesis about the binding affinities to the enzyme DHFR.

In our experiment, the subjects were going to find the best binding structure from a random configuration, which is far from the binding position. There are two different measures here to show the precision of the subjects' results.

1. **Hydrogen bond.** An important index as to how well the subjects have done is the number of hydrogen bonds formed, which almost determines the drug's configuration, and accounts for the major part of the binding energy. In the case of DHFR and inhibitors, the hydrogen bonds account for more than 80% of the interaction energy. A hydrogen bond is usually modeled as two atoms with charges of different polarity, with the separation inbetween determining the binding energy of the two atoms. If all the necessary hydrogen bonds are well formed, the configuration is roughly determined.

2. **Displacements between reference and resulting structure.** The best reference structure is the crystal structure. Displacement (correlation) of resulting structure as compared to the reference structure is another indication of how good the modeling tool is.

Although the hydrogen-bonds should be fixed in place, the other parts of the inhibitor still have some degrees of freedom. As a result, the group that contributes only a small fraction of the binding energy is the most uncertain group in modeling, thus introducing bigger displacements from the crystal data.

Try to imagine that two ends of a chain of atoms are to be fixed in space, but the middle part of it can be flexible. There are many possible configurations, and the average displacement may be big. There is another error-factor about these unrefined crystal structures from 2.8-Angstrom resolution electron-density maps. The original electron density map shows that the peptide chain leading to the carboxy group in inhibitors 309 and 91 is virtually a "column of cloud," as Kuyper puts it [Kuyper 89].

The average displacements from the crystal structure for drug 309 are around 1.5 Angstrom, and range from 1.12 Angstrom to 2.58 Angstrom. The average displacements from the crystal structure for drug 91 are around 1.8 Angstrom, and range from 1.6 Angstrom to 2.2 Angstrom. The above data show better results than those of the Ellipsoid algorithm [Billeter 87]. Billeter *et al.* used methotrexate (MTX) as a test case, and used DHFR as enzyme. The binding of methotrexate to DHFR is similar to our trimetroprim analogues, and the number of atoms in MTX is 54, exactly the same as for compound 91. They selected three rotatable bonds in MTX and made them to be zero degree initially. Billeter's results show that the lowest displacement is 2.4 Angstrom out of all test runs that did converge (only 60% of the runs converged). Although the test cases are limited, the man-computer interface results are better than the Ellipsoid algorithm alone.

The crystal structure of 309 shows that it has four major hydrogen bonds (Table 5.1, Figure 5.12). The fifth and the sixth are also possible hydrogen bonds; however, I do not count them in the following evaluation. Hence for 12 subjects, there should be 48 well formed hydrogen-bonds of these types.

Hydrogen-bond	Separation in Angstrom
1	1.9
2	1.8
3	2.17
4	2.77, weakly formed

Table 5.1: Hydrogen-bond separation for inhibitor 309.

The results of the twelve subjects show that

1. There are 43 (out of a total of 12 subjects*4 hydrogen-bonds = 48) well formed hydrogen-bonds, and five bonds slightly exceed the normal distance of 3 Angstrom.

2. There are no bad van der Waals contacts.

3. The orientations of the base ring and the benzene ring in the resulting drug conformations agree with the crystal structure.

The crystal structure of 91 shows that it has four major hydrogen bonds (Table 5.2, Figure 5.12). The results of the twelve subjects show that

Hydrogen-bond	Separation in Angstrom
1	1.89
2	1.87
3	2.12
4	2.37

Table 5.2: Hydrogen-bond separation for inhibitor 91.

1. There are 48 (out of 48) well formed hydrogen-bonds.
2. There are no bad van der Waals contacts.
3. The orientations of the base ring and the benzene ring in the resulting drug conformations agree with the crystal structure.

From the above analysis, the current ARM system is doing better than those using the Ellipsoid algorithm, both in terms of well formed hydrogen bonds and in displacements.

Test drugs with implicit inhibitor-enzyme-complex crystal structures

Two inhibitors, folate and 301, do not have crystal structures bound to DHFR, but we have implicit information as to the most probable crystal conformations. These two inhibitors demonstrated the capability of the ARM system in modeling an unknown inhibitor-enzyme complex.

Inhibitor 301 is called Pyritrexim, and is an anti-cancer drug currently used in clinic trials. Folic acid, also called folate, has a lower binding affinity than all the other inhibitors.

I used a self-consistency check for these two test drugs, 301 and folate. The results from twelve subjects showed similar conformations within themselves, with the same hydrogen-bond types. Subjectively, when all possible hydrogen bonds were displayed, all twelve subjects believed that the results were satisfactory to their knowledge.

Test drugs without any information of drug-enzyme complex crystal structure

This is the real purpose of a modeling tool like ARM. However, this is also the part where experimental design is very difficult. Only observation is possible. Any single case counts, and any successful exploring is by itself exciting to the chemists.

Stewart from Burroughs-Wellcome Co. used the ARM system for some exploration, and he reported that the ARM system was a fast way of verifying different hypotheses [Stewart 89]. He is interested in a particular enzyme-inhibitor complex.

The protein enzyme is aspartic proteinase from Rhizopus chinensis, and the inhibitor is a reduced peptide (code number u-705-31-E) [K. Suguna *et al.*]. The enzyme aspartic proteinase is found in the AIDS virus. So, if the reduced peptide inhibitor can be modified into a different and even stronger inhibitor, there is chance of inhibiting the normal function of aspartic proteinase. This might lead finally to the death of the AIDS virus. Stewart's research is still in the early stage, and his results are proprietary to the Burroughs-Wellcome Co.

5.3.2 Comments from subjects and pioneer users

The practice/training session lasted for three weeks, with one subject per day. I got fewer and fewer suggestions (a good sign) during the training, and more and more favorable comments toward usefulness, because I collected their suggestions and implemented some of them in a separate demo program, and I was more prepared for the demonstration. None of them said that the ARM system is useless; most of them wanted to "have" similar systems in their lab. The following is an abstract of observations.

A subject who had done molecular modeling using AMBER (a gradient-based local energy minimizer): "This is a fast way to test and verify many different hypotheses within a short time. Most of the time I was not sure the results from AMBER were *the best* results, although people published papers according to the AMBER results. Now I see a fast way to guide the AMBER computation."

A subject who had done molecular dynamics: "The results from the docking system are very good starting points for further molecular dynamics."

A subject who had done laboratory work in X-ray crystallography, but not molecular modeling: "Well, instead of seeing the X-ray crystal structure, I could feel the molecular

forces. This is similar to my work using plastic molecular models (sticks and spheres), and yet it tells me why the drug does not bind in the other way, although there is no geometric hindrance in that configuration."

After two hours of standing in training (5 minutes' rest every 30 minutes), the subjects were asked if they were fatigued using the ARM. Most of them (including 4 females) said "No, not at all, it's fun." No one felt fatigued. Some subjects suggested that a desk-top mini-ARM would be better, so that the screen might match the ARM space.

During the training, when asked if they consciously knew that they were using the force cues from the ARM, some of the subjects said they had not realized that they were using force feedback at all, until I cut off the power to the ARM. Then they felt something was wrong. It appeared to me that when they were actively doing the molecular docking, the force feedback became part of their natural ability.

5.3.3 Can man-computer interaction beat a computer?

In our experiments, subjects using our system can solve practical research problems in 30 minutes. Because of the size of our test drugs, a brute-force systematic search has to deal with the following parameters:

1. 3 D.o.F in translation, 3 D.o.F in rotation.

2. N D.o.F in rotatable bonds. If N rotatable bonds are linearly connected, any bond rotation will affect the other bonds. For most practical inhibitors, the number N varies from 6 to a number depending on the inhibitor.

The time complexity of brute-force systematic-search would be $O(x^{(N+6)})$, where x is the time for 1-D search. Considering the exponential complexity, brute-force systematic-search is out of the question, except for trivial cases involving two or three rotatable bonds in state-of-the-art main-frame computers.

Better algorithms are always possible, but there is no obvious one to date. My belief is that a man-machine interface (visual and force display) in guided computation will always provide a natural way to cut the search space tremendously, especially in the early search phase. Moreover, and conceivably more importantly, the exploring itself yields important insights for the user.

5.4 Conclusion

Force display helped significantly in the 6-D rigid-body manipulation. The speedup is about 30%, instead of 100% as in the previous 6-D simple docking of 6 springs, where there is only one single energy-minimum, and no thinking is involved. In bond rotations, because it is 1-D motion in nature, the addition of force feedback (indirectly from the hand-controller, but not directly from the dials) did not help as compared to visual display alone.

It was demonstrated that visual cues alone are sufficient in molecular docking, although slower. Because biochemists can do molecular docking without using a force system, the visual docking software tool, without the ARM, is valuable by itself and can be distributed to many other laboratories.

On the other hand, even if the use of force display (ARM) is expensive, its performance in 6-D rigid-body manipulation helps significantly ($p < 0.05$) in molecular docking. Furthermore, it is possible that the results may be generalized to other applications, thus introducing a natural man-machine interface.

From my observations, the subjects using the GROPE-III system, with or without force feedback, are getting more precise results than does the Ellipsoid algorithm (one of the best algorithms to date), both in well formed hydrogen bonds and in displacements. It is my belief that man-computer interaction can solve molecular docking problems more efficiently than algorithms alone.

method / subject	F	NF	mean
S1	2238 sec.	1271 sec.	1754 sec.
S2	2098	1754	1926
S3	1297	1307	1302
S4	1143	2134	1638.5
S5	1279	1182	1230.5
S6	598	1411	1004.5
S7	1317	1840	1578.5
S8	1360	2181	1770.5
S9	1086	1630	1358
S10	2214	1197	1205.5
S11	1041	1564	1302.5
S12	1123	771	947
mean	1399.5 seconds	1520 seconds	1430 seconds

Source	df	SS	variance	F-ratio
Treatments	1	87,242	87,242	0.37
Subjects	11	2,179,308	198,118	
T x S	11	2,602,543	236,595	
Total	22	4,869,093		
$p > 0.05$				

where df is degree-of-freedom, SS is sum of deviations, and
T x S is treatments-by-subjects interaction.

Figure 5.9: Performance data for F and NF in molecular docking, both 6-D rigid-body manipulation and flexible chemical bond rotations.

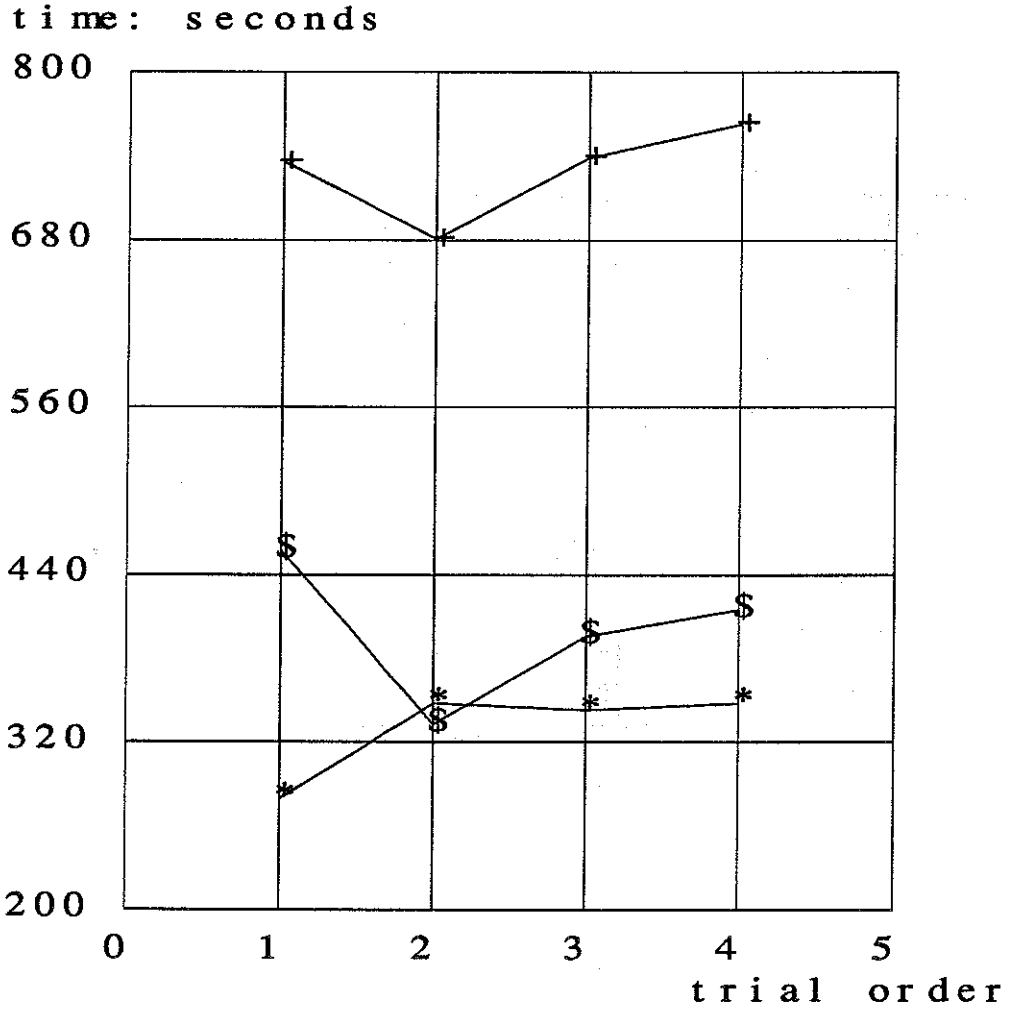


Figure 5.10: The task-completion times as a function of the trial order, where "*" stands for 1-D rigid-body manipulation, "+" stands for overall docking times, and "$" stands for 1-D bond rotations.

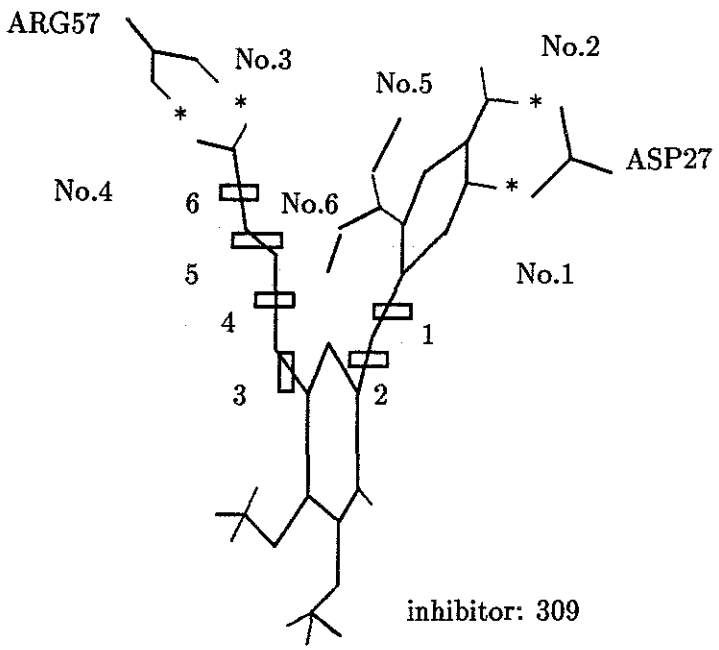
method / subject	F	NF	mean
S1	124	251.7	187.9
S2	244.5	296.1	270.3
S3	106.4	115.7	110.1
S4	123	85.1	104.1
S5	84.2	194.5	139.4
S6	72	116.3	94.2
S7	119.7	214	166.9
S8	295.7	401.6	348.7
S9	203.5	214.5	209
S10	247.9	248.7	248.3
S11	185.6	373.4	279.5
S12	168.7	277.5	223.1
mean	164.6 Angstroms	232.4 Angstroms	198.5 Angstroms

All units are in Angstroms.

source	df	SS	variance	F-ratio
Treatments	1	27,601	27,601	12.99
Subjects	11	50,978	4,248	
T x S	11	23,376	2,125	
Total	22	190,033		
$p < 0.01$	critical value = 9.65			

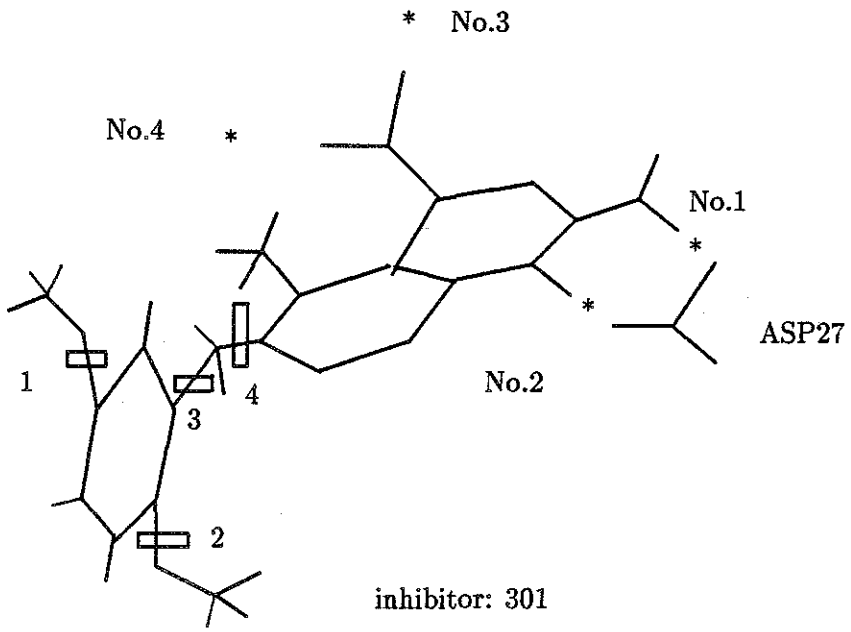
Where df is degree-of-freedom, SS is sum of deviations, and
T x S is treatments-by-subjects interaction.
The path of a molecule is defined as the distance traversed
through translations and rotations.

Figure 5.11: Path length for F and NF in molecular docking.



▭ means rotatable bond, * means hydrogen bond.
ASP27 and ARG57 are residues of DHFR.

Figure 5.12: Inhibitor 309 bound to DHFR in crystal conformation.



means rotatable bond, * means hydrogen bond.

ASP27 is a residue of DHFR.

Figure 5.13: Inhibitor 301: predicted crystal conformation bound to DHFR.

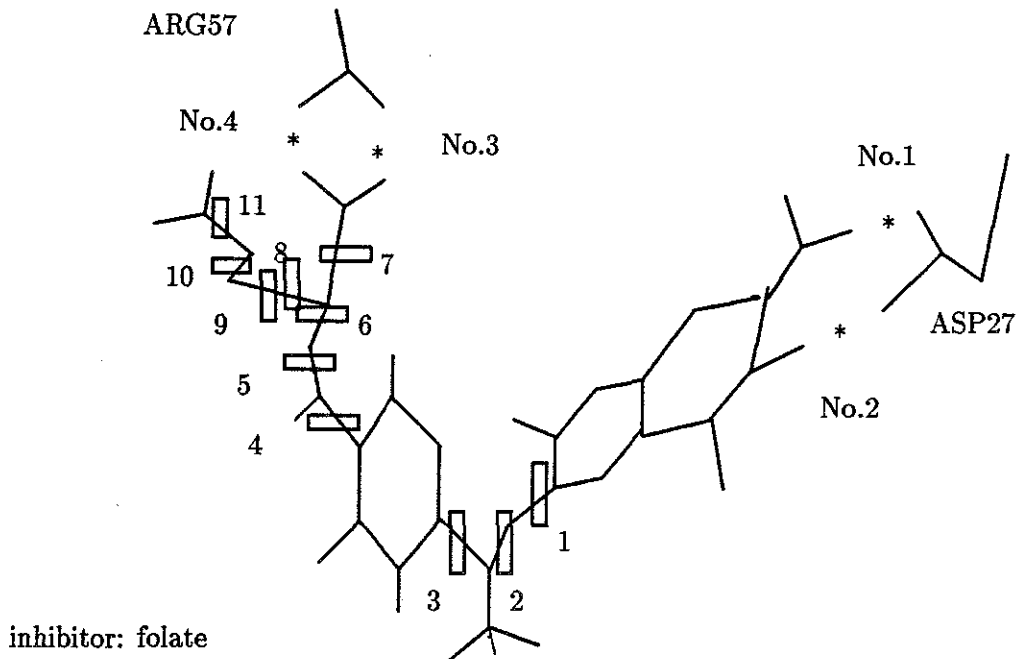


Figure 5.14: Inhibitor folate: one predicted crystal conformation bound to DHFR.

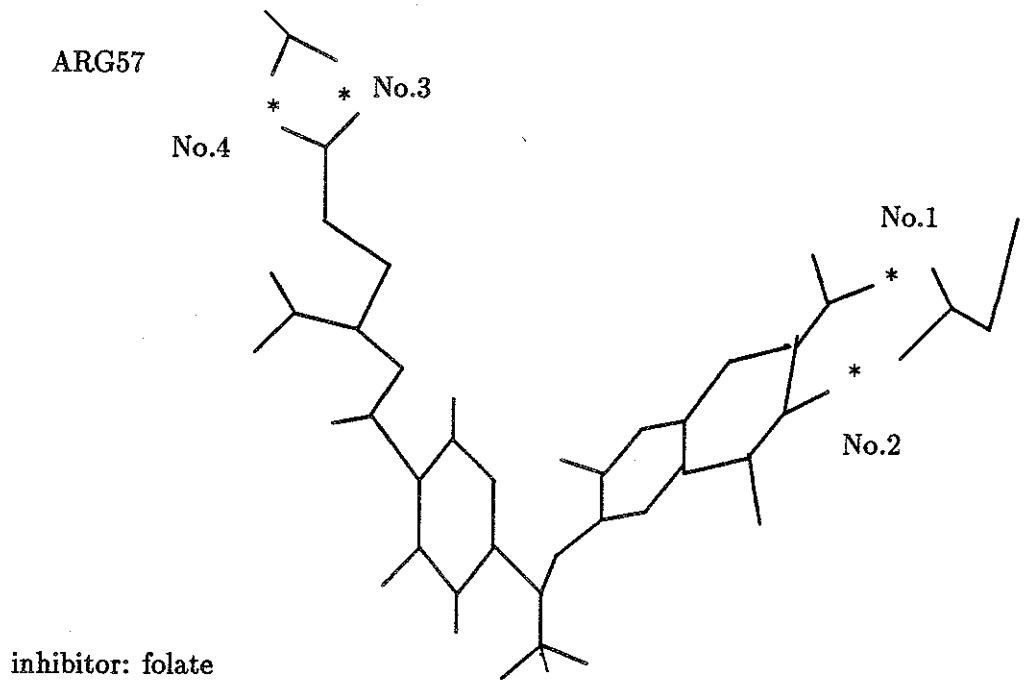


Figure 5.15: Inhibitor folate: another predicted crystal conformation bound to DHFR.

Chapter 6

Creating an Illusion of Feel: Control Issues

Force display is a man-machine interface system that can synthesize forces to the human kinesthetic perception as representations of forces in a virtual world. Creating an illusion of feel by force display adds another dimension to a virtual world, but a man-in-the-loop system poses problems both in design and application.

I investigated a variety of control issues by

1. Analyses in control theory— the stability conditions among sampling period, mass, stiffness, and viscosity in various simulations. These analyses address the question: What is the required system updating rate for system A doing simulation B? More specifically, what is the required sampling rate and latency for the system to be useful for molecular docking?

2. Measurements on the ARM. The experimental data support my predictions from theory.

3. Measurements on the human arm. I followed Hogan's approach and found that there is a significant difference in human arm impedance between radial motion (forward-backward) and tangential motion (lateral) when holding a joystick/hand-controller [Hogan 88].

4. Measurements on Minsky's force-feedback joystick system. Because the joystick system has a fast sampling rate (1200 Hz), I used it to conduct several interesting experiments, and used the analyses in control theory to explain these strange phenomena (which are against our intuition).

6.1 Introduction

In Chapter 2.2.3, "Problems of a man-in-the-loop design," I described typical problems of a force display system. In Chapter 2.2.2, "An example of instability due to sampling", I illustrated an unstable spring-and-mass system. In this chapter, I am going to attack the above problems by using analyses in control theory.

6.1.1 Impedance control theory

The following analysis follows the presentation of the impedance control theory introduced by Hogan [Hogan 87]. Suppose the joystick/hand-controller has mass m, stiffness k, and viscosity b. Define position as x, velocity as v, acceleration as a, the force generated by the motor controlled by a computer as F_s, and force measured by a force sensor as F_{ext} (Fig. 6.1).

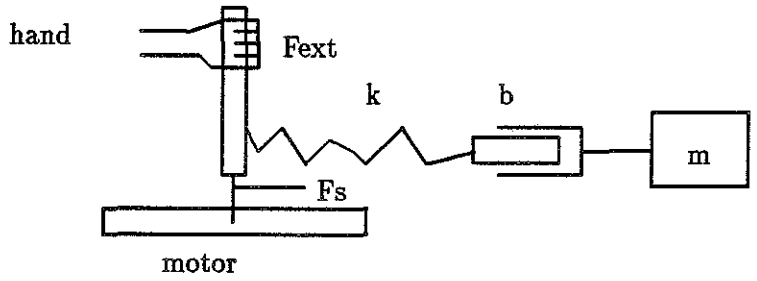


Figure 6.1: A joystick system.

$$ma + bv + kx = F_s - F_{ext} \tag{6.1}$$

Suppose the desired virtual spring-mass system has mass M, stiffness K, and viscosity B, then the force measured at the sensor is

$$-F_{ext} = Ma + Bv + K(x - x_0) \tag{6.2}$$

where x_0 is the rest position. From 6.1, the force required at motor is

$$F_s = ma + kx + bv + F_{ext} \tag{6.3}$$

from 6.2, let $1/M = W$

$$a = W[K(x_0 - x) - Bv - F_{ext}] \tag{6.4}$$

substituting 6.4 into 6.3

$$F_s = mW[K(x_0 - x) - Bv] + kx + bv + F_{ext}(1 - mW) \tag{6.5}$$

Equation 6.5 says that if the position x, velocity v, and the force from sensor F_{ext} can be measured, the system can simulate any object by controlling motor forces only.

6.1.2 Methods in creating an illusion of feel

In the paper we give five cases, and the corresponding strategies for simulating mass, stiffness, and viscosity.

Case 1. The goal is a mass M only, with K = 0, and B = 0. Let M = 10kg, m = 0.01 kg, k = 0.01 N/m, b = 0.01 N-sec/m

$$\begin{aligned} F_s &= kx + bv + F_{ext}(1 - mW) \\ &= kx + bv + 0.999 * F_{ext} \\ &\cong 0.999 * F_{ext} \end{aligned}$$

This says that the force that should be generated is equal to the measured external force. This is what it feels like when we push a rock! Newton's reaction law says that the reactive force is equal to the force applied (F_{ext}). What if we don't have the measured force F_{ext}? Since $F_{ext} = M * a$, and $a = dv/dt$, $F_s \cong 0.999 * (dv/dt) * M$.

Case 2. The goal is a strong spring with stiffness K = 1000 N/m, neutral position x_0, but with small mass M = 0.01kg (M = m).

$$\begin{aligned} F_s &= mW * K(x_0 - x) + kx + bv + F_{ext}(1 - mW) \\ &= mW * K(x_0 - x) + kx + bv \end{aligned}$$

Since $K >> k$, and if v is small, $F_s \cong mW * K(x_0 - x) = K(x_0 - x)$. Therefore, the force generated by the motor is directly related to the target spring stiffness K.

Case 3. The target is a hard surface, like a wooden table. We can approximate a hard surface by a spring with large stiffness, or a big mass M sitting on the ground. It is easy to see that the two approximations have already been covered by case (1) and (2) above. However, case (1) involves the measured F_{ext}, or the acceleration a. In the case that acceleration is approximated by dv/dt, noise is introduced; therefore, the simulation result is not smooth.

Case 4. The goal is a mass M attached to a spring K with viscosity B.

$$F_s = mW[K * (x_0 - x) - Bv] + kx + bv + F_{ext}(1 - mW) \tag{6.6}$$

If we does not have measured force F_{ext}, one can substitute it into $-F_{ext} = Ma + Bv + K(x - x_0)$.

Case 5. The goal is to simulate a spring with stiffness K, mass M = m, and no viscosity. Instead of doing the detailed simulation in the ideal case, we choose to use a very simple method: let the joystick synthesize the spring force based on the position feedback only. The question becomes, what does the human arm really feel?

Let $F_s = K(x_0 - x)$ in our simulation. From Eq. 6.5, assuming joystick stiffness (without power) is 0,

$$\begin{aligned} -F_{ext} &= -[K(x_0 - x) - ma - bv] \\ &= ma + bv + K(x - x0) \end{aligned} \tag{6.7}$$

The true behavior of the system, and so the feel to the human arm, is like holding a spring with stiffness K, mass m, and viscosity b. Although this is not exactly the target spring system (mass m, stiffness K,viscosity 0) in simulation, it is, however, close to the target system if viscosity b is small.

With this simple approach, we successfully built a molecular docking system using the ARM (let $F_s = \sum f(x_i); x_i$ is the position of atom i, $f()$ is a molecular force field function).

Similarly, Minsky and Oliver implemented a "Sandpaper environment," which has patches of textured surfaces [Minsky 90]. Each patch is associated with its visual representation, a pattern or a bitmap. For example, a patch representing a finely grooved surface may have parameters such as the height and spacing of the grooves. Minsky uses a 2D force-feedback joystick, and the force function is: $Fs = g(x, y)$, where x and y are 2D coordinates of a surface, $g()$ is a gradient function on the gray levels in a patch.

6.1.3 Contact instability and the human arm

Ideally, a computer-controlled joystick/hand-controller can simulate any target dynamics. However, in practice almost all systems have *contact instability* problems near a wall (a hard surface). There are several reasons.

1. If a digital computer is used in simulation, sampling delay can make a stable system unstable.

2. If one does not have measured external force F_{ext}, and approximates it with a velocity derivative, more noise are introduced.

3. The two different locations for sensor and actuator cause an instability problem (the non-colocation problem). The dynamics of the link (for eaxample, the lower-arm in the ARM) itself are usually not properly modeled; the link is not a point mass, but a distributed mass [Colgate 89].

Of course, one can make the system stable by adding extra viscosity, or by reducing the stiffness of a simulated hard surface. The former makes the human feel resistance and sluggishness even in free space, whereas the latter makes the hard surface spongy.

To make the problem even more complicated, the system is far from linear. The human operator's own physical characteristics are involved in the feedback loop in exploring the virtual world, and he changes those parameters dynamically and radically. Lanman reported human elbow stiffness to vary from a minimum of about 1.4 N-m/rad to a maximum as high as 400 N-m/rad [Lanman 80]. Cannon and Zahalak's measurements showed that both the limb's natural frequency and damping ratio vary with muscle activation [Cannon 82].

Theoretical analysis [Murray 88] showed that a second-order model with parameters varying with muscle activation and elbow angle was unable to reproduce experimental observations. A simplest competent characterization required a fourth-order model. A fifth-order model was used by Hannaford [Hannaford 89].

Hogan has experimental data to show that a human arm can be accurately modeled as a passive object with constant impedance for periods of up to 1.2 seconds [Hogan 89]. That is, it takes that long to change muscle impedance, rather than the 200ms neuromuscular response time one might have expected.

We followed Hogan's approach and found that there is a significant difference in human arm impedance between radial motion (forward-backward) and tangential motion (lateral) when holding a joystick/hand-controller. There are conditions (both in a hard surface simulation and in pure spring simulation) when the radial motion is always stable whereas the tangential motion is always unstable.

All these data make satisfactory hard-surface simulation unlikely. But there is good news. First, multiple-sensory illusions (vision, force, sound) reinforce each other. Second, even though the system may become unstable during the simulation, the human operator can compensate or avoid it.

1. Multiple-sensory illusion.

   (a) In molecular docking, when one atom bumps into another, a flashing vector is shown between these two atoms [Ouh-young 88]. These discontinuous visual cues help the operator to imagine that he is contacting a hard surface, even though the surface is really spongy.

   (b) Minsky showed a variety of surface textures on the screen while simulating a feeling analogous to haptic perception of texture. Many subjects believed that they were touching the real surface of a concrete wall, sandpaper, or icy ground [Minsky 89].

   (c) Kilpatrick implemented sound feedback by generating a click sound when the hand-controller hit a table [Kilpatrick 76]. Even at a sampling rate as low as 14 Hz, he was able to make subjects believe the table was there and hard.

2. Even though the system may become unstable during the simulation, the human operator can compensate for it or avoid it. For example, when the system is about to oscillate, the human operator first tries to stiffen his arm/hand in order to keep the joystick stationary. At the same time, more viscosity from the human arm adds to the system to make it a damped motion. A human operator has at his hand the following tools to use, although it may take as long as 1.2 seconds (Hogan's estimation) to change them:

   (a) Stiffness ranging from 2 N/m to 800 N/m [Lanman 80]

   (b) Effective mass ranging from 0.2Kg (wrist, tangential motion) to 2.0 Kg (radial motion). One moves his arm forward and backward in radial motion, and the upper arm is involved in a moment of inertia. One moves his wrist and arm laterally by turning the upper arm as a pivot-axis.

   (c) Viscosity ranging from 3 N-sec/m in tangential motion, to 15 N-sec/m in radial motion. Hogan measured an average of 5 N-sec/m [Hogan 89]. However, one cannot change stiffness and viscosity independently. It seems that the human arm damping ratio, $B/(2\sqrt{KM})$, is kept approximately constant (B is viscosity, K is stiffness, M is mass).

3. When the human fails to compensate for system instability, he avoids the instability by using the following strategies:

   (a) Keep away from the instability region by moving away from the current position. This is a natural response, and we observed it a lot in our molecular docking experiments. Whenever an atom bumps into another atom, there is an equivalent hard-surface contact. The human operator quickly learns to move away from the boundary while remembering the boundary position.

   (b) Reduce the force-feedback gain parameter by turning a dial. This is equivalent to making the hard surface spongy or soft.

   (c) If all other means fail, turn off the motor power. In the ARM system, there is a safety foot-switch; unless one step on it, no force comes out from the manipulator, so one can turn the motors off simply by lifting the foot.

6.2 Analysis

What will be the behavior of the system when a human arm is combined with it? Assume that the human arm can be modeled by a second-order system with mass Mh, stiffness Kh, and viscosity Bh. If the arm does not generate forces, the system dynamics equation becomes

$$(Mh + m)a + (Bh + b)v + (Kh + k) * (x - x0) = 0 \tag{6.8}$$

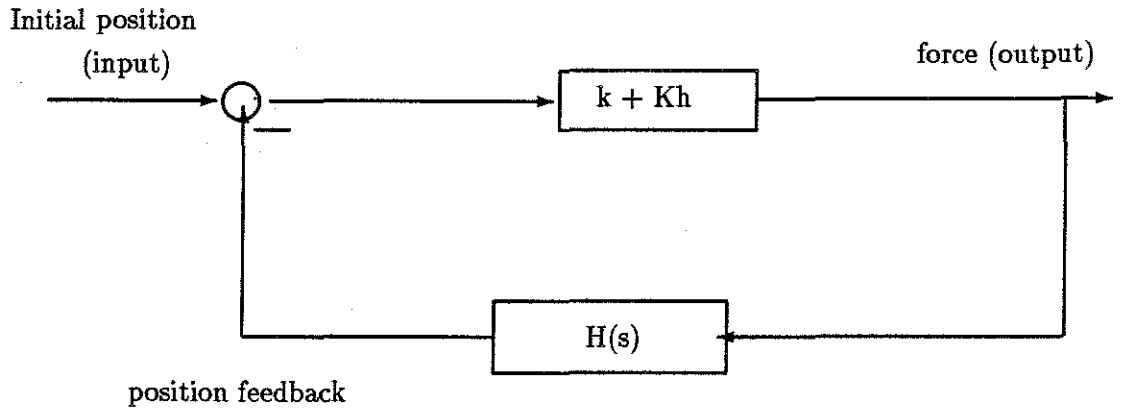
where a = acceleration, v = velocity, x = position, $x0$ = initial position, m = mass of joystick, b = viscosity of joystick, and k = stiffness of a virtual spring. The natural frequency f of the system is given by $2\pi f = \sqrt{(Kh + k)/(Mh + m)}$.

Eq. 6.8 can be Laplace-transformed to accord with a state diagram in Figure 6.2, where the joystick and the human arm are pushed by two springs $(Kh + k)$, with position as the only feedback.

6.2.1 Delayed analog analysis

One way to predict the dynamics of Figure 6.2 is to use the analog control theory, adding a time-delay in the feedback loop. We give a simple analysis of a spring-mass system, with the human arm not included.

Even though delay is introduced, this is still an analog controller. However, we use this model to get some insights before going on to the complicated digital controller.



$$H(s) = 1/[(m + Mh)s^2 + (b + Bh)s]$$

Figure 6.2: Position feedback system.

Assuming that the delay T multiplied by natural frequency s is small, say less than 0.1, e^{-sT} can be approximated by second-order Taylor-series expansion:

$$e^{-sT} = 1 - sT + 1/2s^2T^2 \tag{6.9}$$

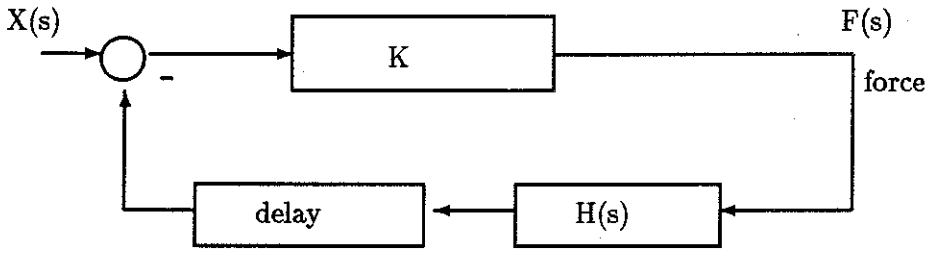
The transfer function becomes

$$\begin{aligned} F(s)/X(s) &= K/[1 + KH(s)e^{-sT}] \\ &= (Ms^2 + Bs)K/[K + (M + 1/2KT^2)s^2 + (B - KT)s] \end{aligned}$$

By the Nyquist criterion, if one of the poles of F(s)/X(s) is located on the right half of the s-plane, the system is unstable. The poles of F(s)/X(s) are equal to zeros of X(s), and are given as

$$[(KT - B) \pm \sqrt{(B - KT^2 - 4(M + 1/2KT^2))}]/(2(M + 1/2KT^2)) \tag{6.10}$$

To be unstable, $KT - B > 0$. This is an approximate solution based on an analog model. There is a constant C involved in this relation, i.e., T > C*B/K, and C is shown to be approximately 2 in more detailed digital simulations.



where K = spring constant,
$H(s) = 1/(Ms^2 + Bs)$, M: mass, B:viscosity,
delay = e^{sT},T: sampling period

Figure 6.3: An analog system with delay T

To understand this solution, suppose B (the viscosity) is small, as in many virtual-world simulations, and K (the spring constant) is big; then the system delay T can easily exceed 2*B/K. A typical example would be that B = 1.17 N-sec/m (joystick), a strong spring K = 400 N/m, and T > 2*B/K = 5.9 ms can cause instability. Even with a loose spring, K = 52 N/m, a delay of T > 2*B/K = 45 ms can cause instability. This places a severe restriction on the force fields that can be simulated by a slow update-rate system.

Surprisingly, the condition for system instability is not related to the mass in this simple analog model.

6.2.2 True discrete model

With a discrete model the solution is not in a closed form, and we have to use simulation to get some insights from it.

Doing so we made the following observations. Let T^* be the maximum sampling period that makes the system stable.

1. T^* is linearly related to 1/K, where K is the spring constant.

2. T^* is linearly related to damping B over a wide range, and then becomes nonlinear (when $B > 16$ N-sec/m).

3. T^* is actually not related to spring mass M when the mass is over a threshold (0.02 Kg). This was a surprise to us, since it means that T^* is not related to the natural frequency.

6.2.3 Details of a digital control model

Figure 6.4 is a digital control model, which describes a way to get a Z-transform from an analog transfer function G(s) (Laplace transform) [Jacquot 81].

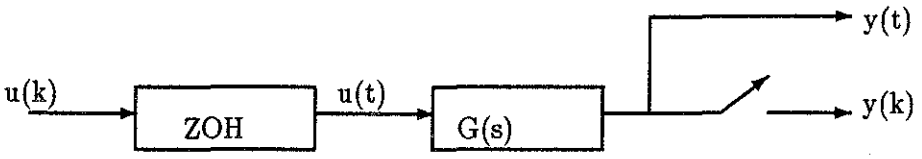


Figure 6.4: A digital model based on sample and hold, where ZOH is a zero-order-hold (sample-and-hold), u(k) is discrete input data, y(k) is discrete output data, u(t) is an analog input signal after sample-and-hold, and y(t) is the analog output signal.

According to the above model, the Z-transform is

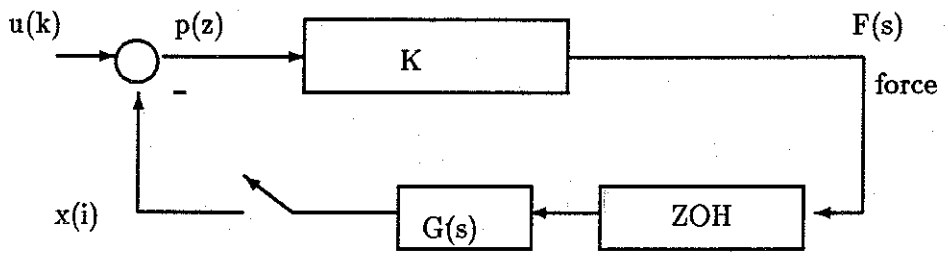
$$\begin{aligned} G(z) &= Y(z)/U(z) \\ &= (1 - Z^{-}1) * Z\_transform(Laplace\_transform^{-1}(G(s)/s)) \end{aligned} \tag{6.11}$$

If one wants to get the Z-transform transfer function of a spring-and-mass system, one can derive it from corresponding Laplace transform results, as shown in Figure 6.5. In this figure, once the input position is sampled, the corresponding output is immediately sent to motors, assuming that the time used in the calculation of forces (a few floating point operations, done in microseconds) is much smaller than that in data acquisition (usually done in milliseconds). This is the way we arranged most of our experiments, except in molecular docking, where computation of force fields is not trivial. In a general model, such as molecular docking, another delay δ ($0 <= \delta < T$, without pipelining) should be added after the box K.

From Eq.6.11 above,
$G(Z) = (1 - Z^{-1})[A/(1 - Z^{-1}) + DTZ^{-1}/(1 - Z^{-1})^2 + C/M(1/[1 - e^{-T*B/M}Z^{-1}])$,
where $A = -M/B^2$, $D = 1/B$, $C = M^2/B^2$, and $C/M = -A$. The closed loop transfer function is

$$p(Z) = K * U(Z)/(1 + KG(Z)) \tag{6.12}$$

According to the Nyquist criterion for the Z transform, if one of the poles of p(z) is located outside the Z-plane unit circle, the system will be unstable.



where $G(s) = 1/[s(B + Ms)]$, M is the mass attached to the end of springs, B = viscosity, K is the stiffness of the springs, $u(k)$ = initial position, $x(i)$ = feedback position of hand-controller, $p(z)$ = deviation from null position used in the spring force calculation, and ZOH is a zero-order-hold (sample-and-hold).

Figure 6.5: A digital control system

The poles of p(z) in 6.12 are the zeros of 1+KG(z). Let $e^{-T*B/M} = W$,

$$\begin{aligned}&[1 + KG(z)] * (1 - Z^{-1}) * (1 - WZ^{-1}) \\ &= (1 + A)(1 - Z^{-1})(1 - WZ^{-1}) + DTZ^{-1}(1 - WZ^{-1}) \\ &-A(1 - Z^{-1})^2\end{aligned}$$

The right side of the above expression can be rewritten as

$$Z2term * Z^2 + Z1term * Z + Z0term = 0 \tag{6.13}$$

where $Z0term = W(1+AK) - KDTW - KA$, $Z1term = -(1+W)(KA+1) + KDT + 2KA$, and $Z2term = 1$. The two roots of the above equation are

$$(-Z1term + / - \sqrt{(Z1term^2 - 4Z0term)})/2 \tag{6.14}$$

6.2.4 Simulation results

Case 1: sampling period T = 0, $e^{-T*B/M} = 1$, $1 + KG(z) = (Z^2 - 2Z + 1)/(1 - Z^{-1^2})$ $= Z^2$

The zeros of 1+KG(z), which are 0.0 and 0.0 (double roots), are located within Z-plane unit circle, therefore, are always stable.

Case 2: Numeric simulation. Assuming M = 0.346 Kg, B = 0.363 N-sec/m, I make stiffness K vary from 52 N/m to 408 N/m; then the maximum allowable sampling period for stability is tabulated, with another term 2*B/K listed for reference.

Stiffness:K Newton-sec/meter	maximum sampling period: T(seconds)	reference term 2*B/K
52.0	0.013916	0.0139
104.0	0.007080	0.0070
208.0	0.003662	0.0035
416.0	0.001709	0.0017
832.0	0.000732	0.00087

From the above table, minimum sampling frequency (1/T) is linearly related to stiffness K, and T is approximately 2*B/K. This constant of 2 was confirmed by an experiment in chapter 7.

The following is the relationship of mass M versus the maximum sampling period T that keeps the system stable, with stiffness = 52 N/m, viscosity = 0.363 N-sec/m.

mass (Kg)	maximum sampling period (seconds)
3	0.0140
2	0.0140
0.5	0.0140
0.1	0.0140
0.05	0.0140
0.02	0.0141
0.01	0.0143
0.0125	0.0145

Clearly when the mass M is above 0.05 Kg, the maximum sampling period allowed is not related to mass. When the mass is less than 0.02 kg, there are variations in the sampling period T.

Figure 6.6 shows the relationship of viscosity to the sampling period T, with mass M = 0.346 Kg, spring constant = 52 N/m. Notice that, in the lower section, the maximum sampling period T appears linearly related to viscosity. When the viscosity becomes big, greater than 8 N-sec/m, the relationship is no longer linear.

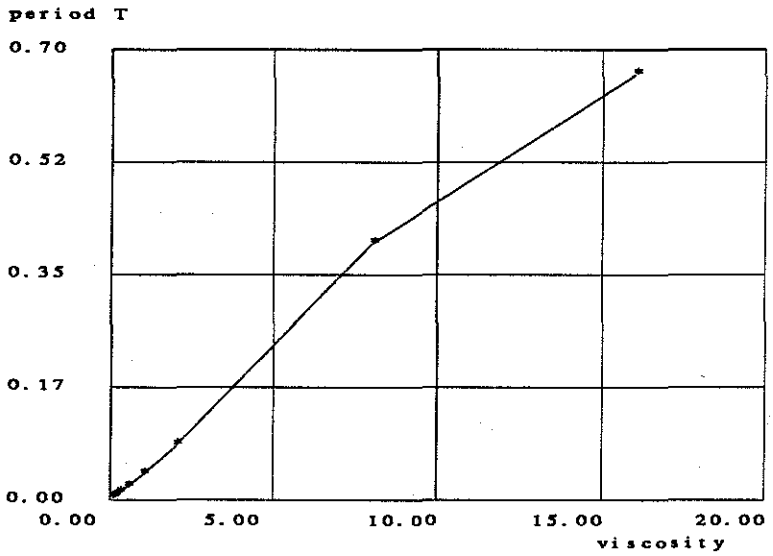


Figure 6.6: A plot of viscosity (in N-sec/m) versus maximum sampling period (in second) in simulation.

6.3 Interesting experiments testing the discrete analysis

The following are interesting experiments constructed to test the results from the above discrete analysis. The notations are Mh (in Kg) for human arm mass, Kh (in N/m) for human arm stiffness, Bh (in N-sec/m) for human arm viscosity, Bs (in N-sec/m) for joystick viscosity, Ks (in N/m) for its stiffness, and Ms (in Kg) for its mass.

6.3.1 Hard surface simulation with different hand motions

Procedure: use the joystick to bump into a hard surface, which is simulated by a spring with stiffness 2773 N/m, with sampling period at 2.8 ms.

Results: the tangential motion is always unstable, but the radial motion is always stable for all thirteen subjects (graduate students in the graphics laboratory).

Parameters used in this experiment.

Ks	Kh	Bs	Bh (tangential)	Bh(radial)
2773	400	1.1	3	10

Explanation: there is a viscosity difference between radial and tangential motion. The stable condition is $T < 2*(Bs+Bh)/(Ks+Kh)$. In the case of tangential motion, the system is unstable since 2*(1.1+3)/(2773+400) = 2.6 ms is smaller than the required sampling period of 2.8 ms. However, in the case of radial motion, it is stable, since 2*(10+1.1)/(2773+400) = 7.0 ms is well above 2.8ms.

6.3.2 Hard surface simulation with low sampling frequency

Procedure: In hard-surface simulation, let the system first run at 1000 Hz, then run at 250 Hz, then run at 100 Hz. Running at 100 Hz, when one bumps into the hard surface, the program increases viscosity to three times the viscosity of the human arm (5 N-sec/m), i.e., Bs = 15 N-sec/m.

Results: At 1000 Hz, the system is stable; at 250 Hz, the system is unstable; at 100 Hz, the system is stable again. At 100 Hz, the subjects "feel" the same hard surface as if the system was running at 1000 Hz.

Parameters in this experiment.

Ks	Kh	Bs	Bh	Bs (within hard surface)
2773	400	1.17	5	15

Explanation: in order to be stable, $T < 2*(Bs+Bh)/(Ks+Kh)$. The system is unstable in hard surface simulation at 250 Hz, since $(5+1.17)*2/(2773+400) = 3.84$ ms = 260 Hz. If the program increases Bs from 1.2 to 15 N-sec/m, three times the human arm viscosity, even the lower sampling rate (100 Hz) makes the system stable, since $(5+15)*2/(2773+400) =$ 12.6 ms = 79 Hz.

The subject did not feel the added viscosity, probably because the added one was still three times that of the human arm's intrinsic viscosity (about 5 N-sec/m). This was a very useful observation, and it shed light on other implementations. In our experiments, the subjects did not feel the viscosity difference; however, it helped tremendously in reducing the required sampling frequency.

We provide one possible explanation for the above observation: the just-noticeable force difference. From Weber's law, the just noticeable difference in force is proportional to the force magnitude, i.e., the bigger the force, the bigger the difference needed to let the human arm tell if there exists a difference. We repeated the experiment by superimposing step functions (pulse width: 1 second, height: from 0% to 20% of the force output) to the ARM lower-arm-roll motor to see if the subjects (only two graduate students were used) could

detect the difference. At about 15% difference in force output, the subjects were able to detect the difference. This may explain why we feel the same hard surface even though it is indeed different as a result of the added viscosity. The particle brake-motor system of [Smith 88] provides a good example of controlling the viscosity, because the brake can have big damping (viscosity) if needed.

Possible conditions when the viscosity can be added without the loss of performance (in terms of human feeling) are:

1. within the hard-surface, which needs geometry information.
2. in any region where the equivalent stiffness is above a threshold (which causes instability at the given sampling rate). This can be implemented as a simple threshold function: if $2B/K > T$, let $B_{new} > TK/2$.

Although the result T < 2B/K appears to make certain simulations difficult, the ability to add viscosity is very promising. Considering that our 6-D ARM has a viscosity of 7.1 N-sec/m, a sampling frequency as low as 30 Hz can still be usable in many applications, since it can simulate a spring of up to 420 N/m without causing instability. When viscosity is added by the program at hard surfaces (simulated by a high stiffness spring: 8*420 N/m), I observed a stable hard surface contact.

6.3.3 Simulation of pure viscosity

If one simulates "stirring a rod in a tank of viscous oil" by $F_s = Bv$, where v is the joystick velocity and B is the desired viscosity, the system equation becomes $m * a + B * v = B * V_0$, where V_0 is the initial velocity. Through similar control analysis, we get the stable condition $T < C1 * M/B$, and $C1 = 2$. Now, if we consider the joystick, M = 0.18 Kg, B = 1.2 N-sec/m, the maximum sampling period for stability is about 300 ms. This is a very long sampling period.

In most of our applications, the viscosity B is kept low, so the system is stable; but the error in velocity derived from position differencing ($\Delta v/\Delta t$) is significant, because the sampling period varies in practice. The human arm can sometimes detect the force difference due to error thus introduced. If one smoothes the velocity by using a low-pass filter, $v = \alpha v_{current} + (1 - \alpha) v_{previous}$, where $0 < \alpha < 1.0$, $v_{current}$ is the velocity at time t, $v_{previous}$ is the velocity at time t-1, the result is much smoothed forces.

6.3.4 The effects of low-pass filters

There are noises in the position input, including the thermal noise of the potentiometer, communication channel noise, and quantization noise (12 bits A/D). If the velocity is calculated from the position difference divided by the sampling period, the velocity term involves two additional noises, i.e., relative error from the difference operation, and error from the irregular sampling period T. Although the irregular sampling period T can be derived from a real clock, the resolution of the clock (for example, 16 ms in Unix) is not usually good enough for such purpose. In practice, the sampling period is assumed to be a constant, and so the irregularity is not accounted for.

Similarly, if acceleration is derived from velocity, the noise is even bigger. One way to deal with these noises is to use low-pass filters to filter out the high frequency noise components. However, adding low-pass filters means additional delay in the sampling period. For example, if a 6-sample-length FIR (finite impulse response) low-pass filter is used, assuming the filtered signal phase-shift is linear, the equivalent sampling period is six times the original one. If the system sampling period is far (six times) less than the critical sampling period, the additional low-pass filter will not cause instability, and, therefore, the input signal is smoothed. This is useful in viscosity simulation, since the sampling period is much smaller than the critical sampling period.

However, in the case of spring and hard surface simulation, the system sampling period is marginally below the critical sampling period, and a six times delay will almost always cause instability. This is very undesirable, and so should not be used.

In one simulation, we try to pull a mass (M) through a connected rubber-band (stiffness K) on a table. Let the force generated be proportional to the rubber-band length deviation, and so only the position of the rubber-band is used in the feedback loop. We observe that when M is big (10 Kg), the system is unstable. So how to simulate heavier mass without causing instability? Since the only feedback term is the rubber-band position, and not the velocity and acceleration of the rubber-band and the mass, it is safe to add low-pass filters to the velocity and acceleration terms, but not to the rubber-band position. This is reasonable, since the delay in rubber-band position caused by the low-pass filter can cause instability in the feedback loop. However, the delays in velocity and acceleration of the mass, as a side effect, make people think that the mass is bigger than before, which is our goal. In short, the above selective filtering creates an illusion (heavy mass) which can fool human beings.

In summary, the possible effects of low-pass filters are (1) smoothed feel, (2) increased sluggishness, (3) increased inertia, and (4) possible instability.

6.4 Two puzzles about the behavior of the human arm

We encountered two puzzles during the study of force display. First, how can the normal human arm be stable, even though the neural-muscular response time is around 200 ms? The puzzle was raised when the hand-controller we used had a sampling frequency of more than 30 Hz and still could easily be unstable. Is it because the human arm has a better way to compensate the system dynamics? The other puzzle was why, even though the joystick sampling frequency was increased from 500 Hz to 1000 Hz, the human arm could still feel the difference in some cases.

In drama and literature, human arms have been portrayed as the wings of a swan, the fists of a bear, the hammer that strikes the bell, and a piece of iron carried by a warrior. These magic tasks of human arms were created by illusions that looked realistic to human eyes. However, the roles of the human arm are quite limited. One interesting result is that, if we assume that the human arm is implemented by a digital controller, the sampling period T must be smaller than 2*Mh/Kh in order to be stable. Typical values of Mh = 0.8 Kg and Kh = 500 N/m show that T must be smaller than 3.2 ms! Obviously this is not a correct model, since the known human neural-muscular response time is much bigger than 1.6 ms, and is around 200 ms.

So, why is the human arm always stable for a healthy person? Hogan coined a term for one kind of controller as *digitally supervised analog control* [Hogan 87]. The idea is that an analog controller can eliminate sampling problems, at the same time allowing some control parameters to be updated by a digital computer infrequently and asynchronously.

Similarly, here we can think of a human arm as an analog controller which is supervised by the mind. But this mechanism is not perfect. Suppose the human arm wants to act like a piece of paper floating in the air, or a piece of iron with large mass, the *digitally supervised control* is simply inadequate. The reason is that, even though the human arm can sense the external force and change the muscular force, stiffness, and viscosity, the time delay is too big to make the arm act like paper or iron. Try letting your hand behave like a piece of paper encountering a striking stick. Although one can see the coming stick by its trajectory and feel its contact with the skin, it is impossible for one to make one's hand act like a piece of paper.

The second puzzle is why, even though the joystick sampling frequency is increased from 500 Hz to 1000 Hz, the human arm can still feel the difference in some cases. Considering that human neural-muscular response time is about 200ms, this phenomenon is hard to explain at first. Our explanation to this puzzle is that although the joystick is running at 500 Hz, it may be unstable at that frequency, whereas it is stable at 1000 Hz. The vibrations caused by this instability can be sensed by the human hand, since there are sensors tuned as high as 400 Hz [Lederman 88]. If the system is stable under both sampling rates (500 Hz and 1000 Hz), we observe that, in a few simulations in the Sandpaper environment, there is no difference in force perception. We hypothesize that the stability in simulation is the major criterion in differentiating force perception.

6.5 Effects of motion scaling

I found that scaling-down in motion sometimes made an inherently unstable system due to low sampling rate become stable. This is a strange phenomenon. I first attempted to explain the above phenomenon using the sampling theory, treating the problem as an aliasing effect in sampling the molecular force field. The sampling theory approach was not successful, especially when the irregular molecular force filed was considered. Although the aliasing effect can be reduced by faster sampling, the sampling theory cannot explain the linear relationship between sampling frequency and spring constant. Failing in making good explanations using aliasing-effect in under-sampling, I used control theory, and was successful.

Regarding the ratio of virtual-object motion (proteins, drugs) over hand-control motion, the ideal case is 1-to-1 motion scaling so that how you move in hand-control space is what you see in the screen space. However, there are cases that the 1-to-1 ratio may be violated. In those cases, many interesting phenomena can be observed, and we can get some insights from them.

Why not keep the 1-to-1 ratio? If one looks at the mouse control in a Macintosh computer or other workstation (SUN, DEC3100, etc.), the scaling of mouse motion to screen cursor motion is always less than 1.0, and there are cases when cursor motion is mouse velocity/acceleration-dependent. This adaptation in scaling indicates that it considers the tasks under manipulation, for example, fast translation to point to a new window, or fine-tuning in VLSI lay-outs and alignments. Of course, the mouse space and the screen space are limited by table (scanner pad) and screen size respectively.

Similarly, the ARM hand-controller space is limited to 2 feet in translation. The screen size of a E&S PS300 screen is 19 inches, and the special projector screen behind the ARM is 4 feet by 3 feet. During the fine-tuning of the drug positions, the subjects need the ability to fine-tune orientations as well as translations. Therefore, two keyboard commands were used to modify the motion scaling: F for faster motion, the object will move faster, S for slower motion, the object will move slower.

Lipscomb observed that human operators did not notice the scaling down in orientation as long as the scaling is fixed, even if the scaling is 2 to 1 [Lipscomb, 87]. Our observations in molecular docking supported his observation. When subjects were highly involved in the task of molecular docking, they did not realize that there were scalings both in orientation and in translation. When asked about if there was any scaling, they said that there was probably scaling-down in translation, but not in rotation. The bond twisting in rotatable bonds was another demonstration of human adaptation to rotations. There were no complaints about whether the dials are mapped in 1-to-1 scaling or 3-to-1 scaling-down.

The reason may be that human beings are well trained and are adapted to using tools with different motion scaling (manual steering in driving, mouse manipulation in computers), and eye-hand coordination is used rather than the kinesthetic feedback of absolute arm positions.

Scale-down in motion has another advantage in force field simulation. Suppose the force field is a linear spring force, with spring constant K. Now a motion scaling down of 3 to 1 means that the effective spring constant is K/3– the hand-controller must be moved three times as far in order to get the same force as before. The critical sampling period T in spring simulation should be smaller than C*B/K, where C is 2, K is the virtual spring constant, and B is the viscosity of the ARM plus human arm. Now, replacing K by K/3, the critical sampling period becomes three times bigger than before, since the viscosity of human arm and ARM stays the same. If the ARM system is running at 30 Hz, and the maximum spring constant that can be simulated is K, when a scaling-down of 3 to 1 is used, the same system can simulate a virtual spring constant of 3*K without causing instability. Of course, what the user really feel is an effective spring with spring constant K, instead of 3K.

Conclusion

We did not use all these theories in designing our first systems. When problems came one by one, we realized that an analysis would be useful. The analysis helped us understand the

performance of our current systems, and we believe it will also contribute to the design of new force display systems.

Chapter 7

Determining the ARM's Mechanical Impedance

7.1 Mechanical impedance of the ARM and force-feedback joystick

We modeled the ARM and the force-feedback joystick as second-order systems, and derived their mechanical impedance by several experiments. The measurements of the ARM are done at the shoulder-rotation joint, since this joint involves the worst case inertia. The measurements of the joystick are done on one of the two symmetrical axes. The stiffness K can be measured by a force scale. The equivalent mass M can be derived from oscillation frequency by $\omega = \sqrt{K/M}$, and the viscosity can be derived from $T = C * B/K$, where B is viscosity, C is a constant, and T is the maximum sampling period for stability. Notice the linear relationship between the maximum sampling period T and $1/K$ in Fig. 7.1. We estimated the ARM's viscosity to be 7.1 Newton-sec/meter. Similarly, the joystick's viscosity was estimated to be 1.17 Newton-sec/meter (Fig. 7.3).

Notice the linear relationship between frequency (in radian) squared (ω^2) and stiffness K in Fig. 7.2. From Fig. 7.2, we estimate the mass of the ARM to be 3.76 Kg. Similarly, the mass of the joystick is estimated to be 0.18 Kg (Fig. 7.4).

Table 7.1 is a summary of the measured mechanical impedance.

Although there is a linear relationship between sampling period and viscosity/stiffness in critical oscillation, I had to determine, by experiment, the constant C in the formula $T = C * B/K$, where B is viscosity and K is stiffness. According to the analysis in Chapter 6, for any stiffness and viscosity, there exists a critical sampling period that makes the system

types	comments	mass	viscosity	stiffness
ARM	shoulder link	3.76 Kg	7.1 N-sec/m	program control
Joystick		0.18 Kg	1.17 N-sec/m	program control
Human arm	tangential radial	0.2 Kg 2.0 Kg	3 N-sec/m 15 N-sec/m	800 N/m (max)

Table 7.1: Mechanical impedance of the ARM, the force-feedback joystick, and human arm

unstable. When the system is initially stable, we can add negative viscosity to the system through program control, until the system becomes unstable. The conditions are

$$B_{T1} + AddedViscosity = B_{T2} \tag{7.1}$$

where B_{T1} and B_{T2} are the viscosities at critical sampling periods $T1$ and $T2$ respectively, and $AddedViscosity$ is the program-added negative viscosity. From Eq. 7.1, substitute TK/C into B,

$$T1 * K/C + AddedViscosity = T2 * K/C \tag{7.2}$$

where $T1 * K/C$ is the ARM (shoulder rotate) viscosity when there is no program-added viscosity, and is $14.2/C$ ($T1 * K$ is from Fig. 7.1).

In the above equation, I use T2 and AddedViscosity as variables in order to estimate the constant C. In Fig. 7.5, the estimated constant is 2.3. The theoretical value of C is 2.0. The difference between 2.3 and 2.0 is perhaps due to two factors. First, the added negative viscosity is derived from the velocity by position difference and is not accurate. Second, the measurement is based on a system's not being critically unstable, but is very unstable (easier to measure the oscillation). In this case, the sampling period $T2$ is bigger than the critical sampling period, and so the constant C is over-estimated (Eq. 7.2).

Different behavior of the ARM and the joystick in simulations. In spring simulation (without the human hand), the critical sampling period must be smaller than viscosity/stiffness*constant in order to be stable. For the same spring constant K, since $T < B/K$, the ARM can be $B_{ARM}/B_{joystick} = 7.1$ (N-sec/m)/1.17 (N-sec/m) = 6.4 times slower in sampling rate than the joystick and still be stable. For example, right now the ARM is running at 60 Hz for the spring simulation, and the equivalent sampling rate for the joystick would be 60*6.4 = 384 Hz. This is confirmed by my experiment. If the hard surface is simulated

by a spring with a very large spring constant, the necessary sampling rate for the ARM can be proportionally slower than that of a joystick.

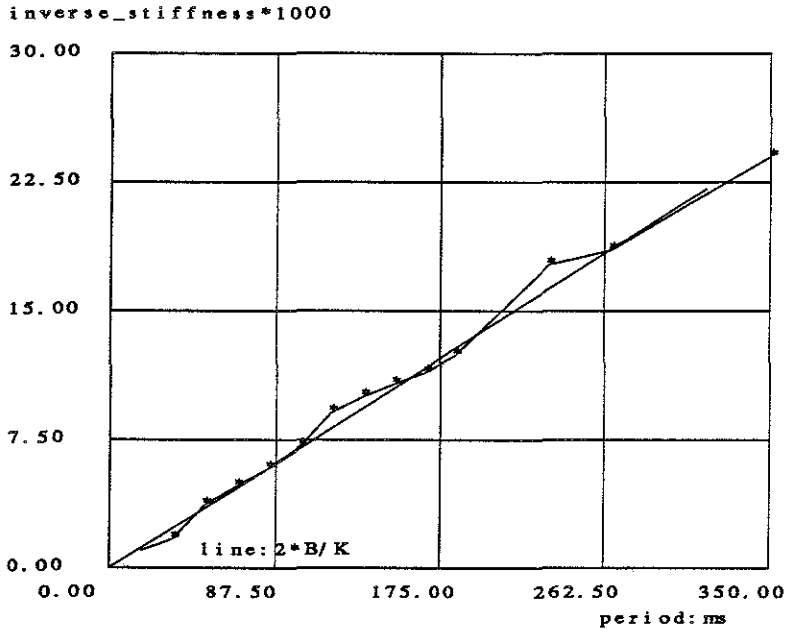


Figure 7.1: ARM's shoulder-rotation 1/stiffness (in m/N) versus sampling period (in ms).

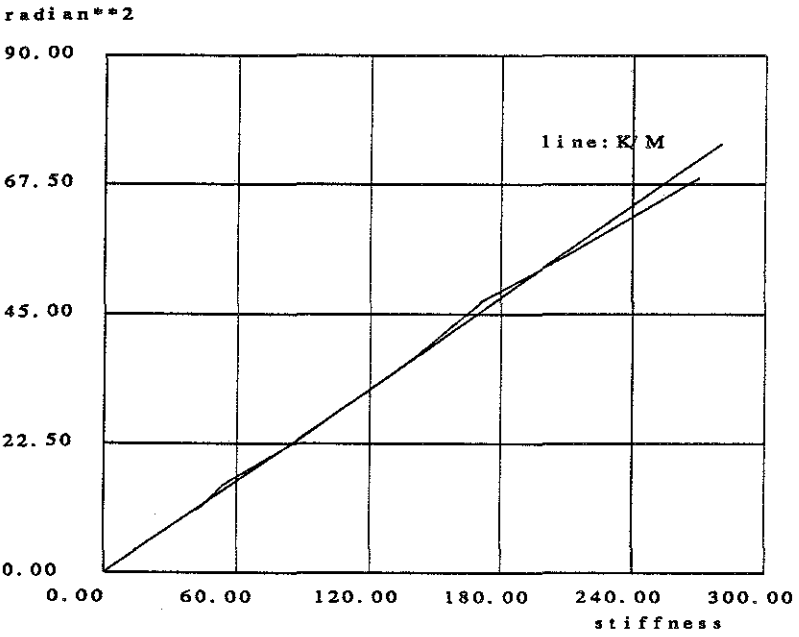


Figure 7.2: ARM's shoulder-rotation frequency squared (in radian) versus stiffness (N/m).

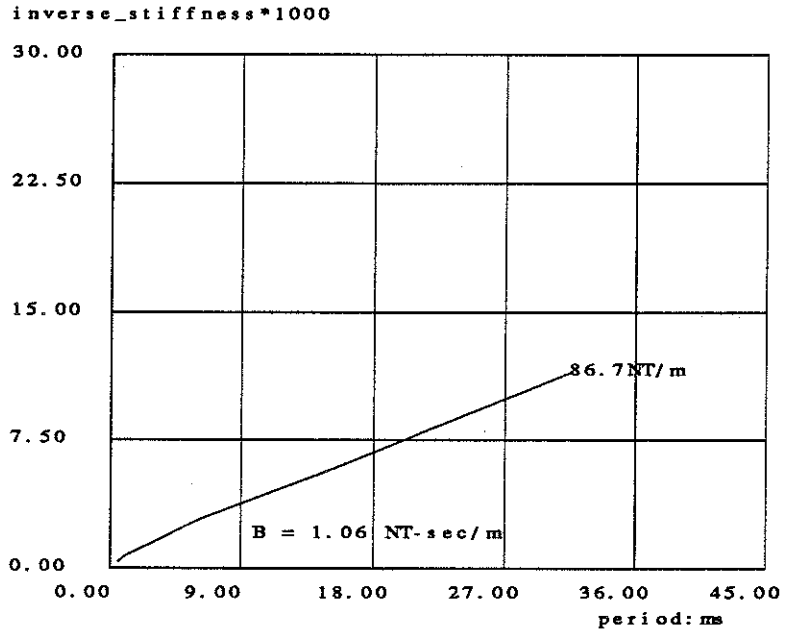


Figure 7.3: Joystick 1/stiffness (in m/N) versus sampling period (in ms).

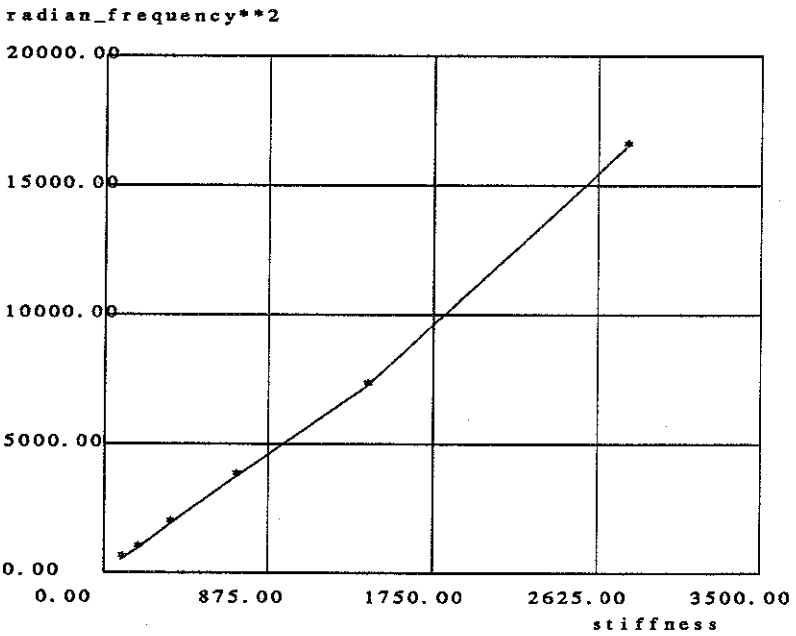


Figure 7.4: Joystick frequency squared (in radian) versus stiffness (in N/m).

stiffness K (N/m)	added viscosity NT-sec/m	sampling period T2 (ms)	estimated constant C
41.7	-5.52	33.3	2.32
41.7	-5.25	50.0	2.31
41.7	-5.23	66.7	2.18
41.7	-4.36	83.4	2.46
41.7	-4.17	100.0	2.36
53.9	-4.36	66.7	2.43
53.9	-3.64	83.4	2.66
53.9	-3.49	100.0	2.52
80.9	-4.8	33.3	2.39
80.9	-4.59	50.0	2.21
80.9	-3.49	66.7	2.54
80.9	-2.54	83.4	2.94
80.9	-2.18	100.0	2.80
93.1	-4.8	33.3	2.31
140.0	-4.22	33.3	2.26
269	-4.5	33.3	2.16

$T1 * K/C + AddedViscosity = T2 * K/C.$

The ARM shoulder rotation viscosity = T1*K/C = 14.2/C,
let $T1 * K$ be fixed at 14.2 when $AddedViscosity$ is 0.

Figure 7.5: Estimated constant C through the addition of negative viscosity.

Chapter 8

Algorithms vs. Mind-Guided Exploration

This chapter describes my efforts and experiences in developing software algorithms that can solve the molecular docking problem without using force feedback. There are many interesting numbers coming out of simulations on a SUN4 workstation. These efforts let me explore several possibilities, including interactive tools. The objective is to develop a docking tool that can be used on workstations, which are accessible to many biochemists.

8.1 Docking by algorithms only

To see why a mind-guided exploration of the space of possible configuration might have an advantage, let us briefly examine the computation-only alternatives.

First, one must step through the configuration space, either by brute-force increment along all its N dimensions in nested order, or by some other path generator that seeks the minimum-energy configuration more efficiently.

The minimizing methods that would seem applicable are:

1. Brute-force search of the entire space. The computation complexity with N degrees of freedom will take a very long time.

2. Molecular mechanics. This will get into local minima. The complexity of molecular mechanics is $O(n^2)$, where n = total number of atoms. In general, n is between 100 and 10,000. For example, if n equals 100, it takes 6 minutes on a VAX/780 to converge to a local energy minimum within 0.1 Kcal in accuracy [Dearfield 88].

3. Simulated annealing using molecular dynamics. This can find the global minimum, but it takes a long time.

4. Quantum mechanics. This will get into local minima. The computation complexity is $O(n^4)$, where n = number of total orbitals. For example, it takes 38 Cray hours to calculate dimethyl phosphate-methyl guanidinium [Dearfield 88].

8.1.1 Brute force search: estimated at about 118 years

To use a brute force systematic search, let us consider the following simplified docking problem. Both the receptor molecule and the drug molecule are rigid bodies without rotatable bonds. To speed up the molecular mechanics calculation, a 3-D tabulation method is used. The drug molecule has only six degrees of freedom.

Assuming that the drug molecule can appear within the bounding box (20x20x20 Angstroms) of a receptor molecule, the search is done every 0.2 Angstrom in translation, 5 degrees in rotation. Given the position and orientation of a drug molecule, a SUN4 can calculate the energy in 10 ms. So, the time to solve the docking problem on a SUN4 workstation is $(20/0.2)^3 * (360/5)^3 * 0.01\ second = 118\ years$.

8.1.2 Simulated annealing: slow annealing process

The next thing I tried was simulated annealing, since this algorithm seems to be similar to the true docking process in nature [Kirkpatrick 83].

The simulated annealing algorithm, instead of following the steepest-descent path, chooses a new configuration by a probability based on the Boltzmann distribution, $p(\Delta E) = e^{-\Delta E/kt}$, where t is the system temperature, k is the Boltzmann constant, and E is the binding energy. The temperature is lowered as time passes. The perturbation of the next configuration is given as a random 6-D vector, and each component of this vector is a random number. This 6-D vector consists of three translations and three rotations. Again, I used the simple model of a rigid-body receptor-inhibitor complex.

My experience is that the temperature function is hard to choose, the running time is measured in hours, and the results sometimes depend on the initial position. Furthermore, once the results are obtained, one can still sometimes get better results by running a gradient-based energy minimizer, and the minimized results do not necessarily agree with the results from several runs of simulated annealing. So, it is a hard problem to find out which configuration is the global energy minimum.

Many researchers have used molecular dynamics (similar to simulated annealing) for decades, but none of them have claimed that they can use it to solve the docking problem. My understanding is that the "annealing process" of molecular dynamics in nature should be sampled in picosecond steps. So, if the process lasts for one millisecond in nature, there are 10^9 steps to be simulated in a computer. Even if each step can be calculated in 10 ms, the total computation time will be 3.8 months on a SUN4 workstation.

8.1.3 Mind-guided exploration: reasonably short time

From the previous section, it is clear that neither brute-force search nor simulated annealing is suitable for workstations. In my estimation, they are not suitable for supercomputers either, since a SUN4 has about 1/12 the speed of a CRAY supercomputer, if there is no vectorization and multiprocessing in the codes.

So, I tried the human-machine interaction approach. My assumption is that, if the human user can bring the drug molecule in the neighborhood (3x3x3 $Angstrom^3$ volume, with 15 degrees in rotations) of the true global energy minimum, the search time by brute-force alone will be reduced tremendously. Similar to the brute-force search approach above, for a rigid-body drug-receptor complex, the computation time for a volume of 3x3x3 $Angstrom^3$, with rotational freedom of 15 degrees (X,Y,Z rotation), will be $(3/0.2)^3 * (15/5)^3 * 0.01$ second = 15.2 minutes. Note that 15 minutes is reasonably short, and a user can have a coffee break and wait for the results.

When the torsional angles (M D.o.F) are considered, one can use either brute-force search, or a gradient-based energy minimizer to find the converged solution after the translation and rotation are fixed.

Visual representation of forces and torques

The question left is how to provide a tool so that a user can do the docking visually, and bring the drug in the neighborhood of the true global energy minimum position.

In chapter 4, I have implemented a visual display of forces and torques. This is of course better than showing six independent vectors (translations and rotations) on the screen. I used this method in my six-springs experiment (chapter 3), and half of the subjects volunteered that they liked it.

When there is only one energy thermometer on the screen, and the users are required to find the energy minimum of six-springs visually, it takes much longer than with my new visual representation. From observations of six-springs experiments, using one energy thermometer alone was actually a frustrating experience, even though the ARM provided direct 6-D inputs.

Visual display of forces and torques helps in six-springs experiments. It is not clear how this new visual representation will help in the true molecular docking, since there are no experimental data. I put a virtual sphere at the center of the drug molecule, and the visual forces and torques are the same as in Chapter 3. Two volunteer chemists tried it and said that there is already visually rich information on the screen (for example, bump checking, hydrogen-bonds, and energy thermometers), and the addition of visual forces and torques at the center of the screen tends to distract their attention. Perhaps a separate display area on the screen for these visual torques will help.

In chapter 5, we have demonstrated that multiple visual cues (energy thermometers, bump checking, hydrogen-bonds, models of molecules) can help a user to find the neighborhood of the global minimum. Therefore, providing many visual cues and still maintaining a visually consistent screen appears to be the correct way to go.

Chapter 9

Software Constructs

This chapter describes the overall software constructs used in molecular docking and two other simulations, namely, six-springs simulation and fishing [Appendix D]. I emphasize the simulation of forces and torques in 3-D, since this is where a force display differs from other visual simulations.

9.1 The main program

My molecular docking software is arranged in a single loop. In general, I want to have a fixed update rate, so that the output forces are smooth. The following are the overall program specifications.

```
-------------------------------------------------------------
                main program    dock.c
-------------------------------------------------------------
   The  control flow is

      initialize_simulation()
      initialize_graphics_engine()
      armopen()   /* open 16 channel A/D and 8-channel D/A */

      while (TRUE) {
         armread1(ad_data)   /*read ARM A/D  data: ad_data[ ] */
         adtoangles(ad_data, angles) /* convert A/D data into
                                        angles[ ] in radian */
         adtoswitches(ad_data, switches)
                           /* convert A/D data into switch
```

```
                          status and dial readings */
      nest_(minusangles,truegrip_in_arm,force,fout,jaco,fpm,ss7)
                        /*obtain the 6x6 Jacobian matrix
                          of the ARM, using the joint angles*/
                        /* a FORTRAN subroutine       */
      evalforce()       /* force and torque evaluator */
      moment_trans()    /*force and moment transformation
                          between coordinate frames */
      action()          /*keyboard command interpreter */
      bumpchecking()    /* collision (bump vector) display*/
      graphics_output() /*update graphics*/
      armwrite1(da_data) /*send control signal da_data[ ] to
                          digital-to-analog converters */
   }
   armclose()  /* close A/D and D/A */
   close_graphics_engine()
   end

where   initialize_simulation() can be one of
      (1) initsocket(gridfname, drugfname);/*Ethernet socket
                                            initialization*/

      (2) initdhfr(molfname);
          initeval(gridfname, drugfname);
          /* molecular docking initialization*/

      (3) initevalstr(); /* read springs coefs */
          /*initialization of six-springs simulation */

      (4) init_fish();    /*initialize fish coefs */
          /*initialization of fishing simulation */


 and initialize_graphics_engine() can be either
      (1) psetup(0)
           /* for E&S PS300 vector graphics engine  */
      (2)  datalink_init();
```

```
                /* for Pixel-Planes-4 raster graphics engine */


    and graphics_output() can be one of

 (1) psdrugupd(force_in_scr, torque, force, drug_in_mol,
          energy, bond_mat)
float   force_in_scr[ ],        /* force vector, screen space */
        torque[ ],              /* torque vector, screen space */
        force[ ];               /* force vector, world space */
MATRIX   drug_in_mol;     /* drug position */
float   energy[ ] ;             /* binding energies */
MATRIX  bond_mat[ ] ;           /* chemical bond rotations */
    /* update the drug position, energy thermometers,
       show force and torque vectors */
    /* In fishing simulation the drug position can be replaced by the
       fishing pole handle position  */



  psfishupd(fish_in_mol)
  MATRIX   fish_in_mol;
  /* display fish position  */


   psviewupd(global_mat)
   MATRIX   global_mat;    /*global matrix */
   /* global view updates*/

(2)  pxplviewupd(mol_in_scr, drug_in_scr )
     /* In pxpl-planes-4: update protein enzyme and drug at the same time */
```

9.2 Forces and torques transformation

Torques in a molecular docking simulation must be calculated according to some reference point in the drug molecule. The drug molecule is manipulated by the user, and the drug-receptor complex is similarly manipulated; the torques have a new reference point during each update, and the forces have a new orientation in screen coordinates. The above transformations need force and moment transformations among coordinate frames. These are derived from the following equations.

Given force F and moment M in coordinate system E, what are the equivalent forces and moments in coordinate system T? Solution: If 4x4 matrix T^* describes the coordinate transformation from frame E to T, and if T^* is expressed as $[N, O, A, P]$, where N, O, A, P are column vectors, then

$$M_x^T = F \cdot (P \times N) + M \cdot N \tag{9.1}$$
$$M_y^T = F \cdot (P \times O) + M \cdot O \tag{9.2}$$
$$M_z^T = F \cdot (P \times A) + M \cdot A \tag{9.3}$$
$$F_x^T = F \cdot N \tag{9.4}$$
$$F_y^T = F \cdot O \tag{9.5}$$
$$F_z^T = F \cdot A \tag{9.6}$$

where the force vector in frame T is $F^T = (F_x^T, F_y^T, F_z^T)$, and the moment in frame T is $M^T = (M_x^T, M_y^T, M_z^T)$; "$\cdot$" means inner product, and "$\times$" means cross product.

It is much easier to provide a software interface that needs the forces and torques relative to the world coordinate origin, without considering the new center of the drug, and let the program do the transformation. In our system, the relevant coordinate frames are given in Figure 9.1, with four steps labeled to give the intermediate stages of force-and-torque transformation.

The following is a step-by-step derivation of what was done in molecular docking, from a routine that calculates forces to the point where the forces are generated by the ARM.

Step 1: a force evaluator calculates the molecular forces and torques, where forces and torques are relative to the protein origin. Notice how this simplifies the force simulation by providing a fixed reference point, which is the world coordinate center. This applies to my six-springs and fishing simulation.

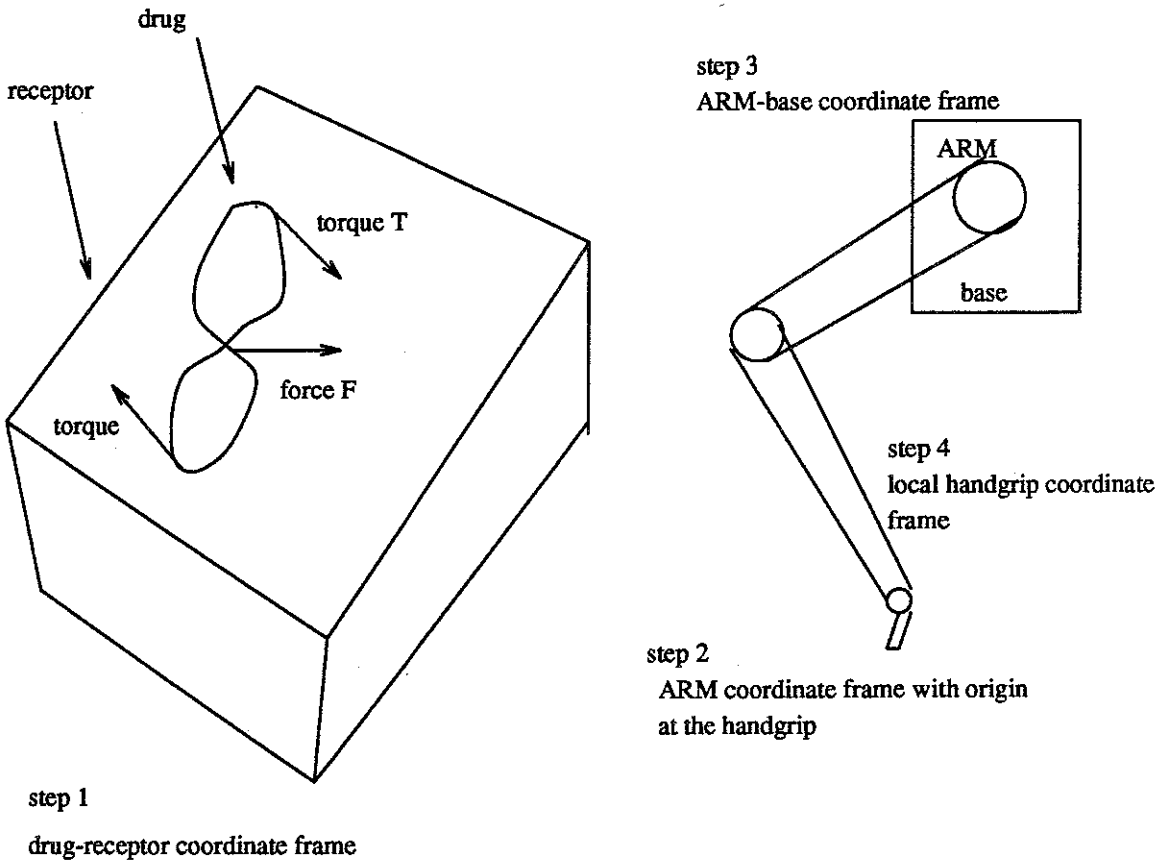


Figure 9.1: Force and torque transformation in molecular docking.

```
-------------------------------------------------------------
    evalforce(&Drug_mol,eflag,fflag,tflag,drug_in_mol,Eforce,
              Etorque,energy,&erpair);
-------------------------------------------------------------
```

where output variables Eforce[] and Etorque[] are relative to the protein origin, which is the world coordinate frame origin. The question is, What are the torques, given a reference point in the drug? This is the torque one feels in molecular docking.

I first provide a routine that can do force and moment transformations between two coordinate frames, assuming there is only translation between these two frames.

```
    The moment_transform() function below will give the transformation
    by calculating
```

```
    force.new = force.old ;
    moment.new = force.old x translation + moment.old ;
                    where x is cross product.
moment_transform(Eforce, Etorque, drug_center)
float  Eforce[ ],Etorque[ ],drug_center[ ];
{
  /* Etorque = Eforce[ ] x drug_center[ ] + Etorque[ ];
     where "x" means cross product                          */
 }
```

Next, the following transform the torques from world coordinate (receptor) origin to drug position as the new origin.

```
        for ( i = 0 ; i < 3 ; i++) {
            Eforce_in_mol[i] = Eforce[i] ;
            Etorque_in_mol[i] = Etorque[i] ;}
        for ( i = 0 ; i < 3 ; i++)
            shifted_origin[i] = drug_in_mol[i][3];
        moment_transform(Eforce_in_mol, Etorque_in_mol, shifted_origin);
```

Step 2: here we get forces and torques in the receptor coordinate frame. What are their values in ARM coordinate frame? Here the handgrip position is the origin.

```
        /* precondition: Eforce_in_mol, Etorque_in_mol     */
        /* postcondition: Eforce_in_arm, Etorque_in_arm   */
        /* make a copy now, since the transformation takes 4x4 matrix
           as input                                                  */
        for ( i = 0 ; i < 3 ; i++) force_in_mol[i][3] = Eforce_in_mol[i] ;
                force_in_mol[3][3] = 0 ;
        for ( i = 0 ; i < 3 ; i++ ) force_in_mol[i][2] = Etorque_in_mol[i] ;
                force_in_mol[3][2] = 0 ;
         /* rotate it, so to get forces and torques in ARM coords */
         f4matmat(mol_in_scr, force_in_mol, force_in_scr);
         /* get forces in screen coordinate system, since I need to display
            force vectors on the screen */
         invarm_matrix(force_in_scr, force_in_arm};
         /* from screen coordinate system to the ARM coordinate system */
         for ( i = 0 ; i < 3 ; i++ ){
                        Eforce_in_arm[i] = force_in_arm[i][3] ;
```

```
Etorque_in_arm[i] = force_in_arm[i][2] ;
} /* copy into  Eforce_in_arm    */
```

Step 3: we have forces and torques in the ARM coords, and the ARM handgrip is the origin. What are the forces and torques in the ARM (shoulder) base frame? The following are transformations from ARM handgrip origin to the ARM shoulder base as new origin.

```
for ( i = 0 ; i < 3 ; i++ )
    shifted_origin[i] = truegrip_in_arm[i][3] ;
moment_transform(Eforce_in_arm, Etorque_in_arm, shifted_origin) ;
/* Express the force and moment relative to the ARM base ! */
```

Step 4: one multiplies the ARM Jacobian matrix by the force and torque, relative to the local ARM handgrip coordinate frame, to get the desired joint torques. Originally the Jacobian matrix describes the differential changes between two coordinate systems, but through the "equivalence of virtual work done in two different coordinate systems," the Jacobian matrix can also describe the forces and torques between two coordinate systems (Appendix A).

$$T = J^{transpose} * F \tag{9.7}$$

where J is a 6 by 6 Jacobian matrix, force vector $F = [F1, F2, F3, M1, M2, M3]$, torque vector $T = [T1, T2, T3, T4, T5, T6]$, $(F1, F2, F3)$ is a force vector, $(M1, M2, M3)$ is a torque vector (in local handgrip coordinate frame), and $T1...T6$ are ARM joint torques. See Appendix A for more details.

Notice that the forces and torques should be expressed in the local handgrip coordinate frame in order to use the Jacobian matrix, and this requirement will cause a lot of trouble just for debugging purpose (Appendix A). The best way is to find a fixed reference frame, for example, the ARM-base coordinate frame.

```
/* scale forces: force_scale is with the force dial      */
 for ( i = 0 ; i < 6 ; i++)
       force[i] = Eforce_in_arm[i]*force_scale  ;
 /* store the new value of input forces and moments for
     nest_() routine to calculate joint torques for each
     joint.  input values are contained in force[]   */
/* A routine for Jacobian matrix calculation */
nest_(minusangles,truegrip_in_arm,force,fout,jaco,fpm,ss7)
```

```
    /* fout[ ] is the force & torque input expressed in the
    current hand_grip coordinate system. If multiplied
    by the Jacobian matrix jaco[ ], the result is the output
    torques for each joint in the arm  */
  for ( i = 0 ; i <=5 ; i++ ) {
        sum = 0.0 ;
        for ( j = 0 ; j <= 5 ; j++ )
              sum += jaco[j][i]*fout[j] ;
        torque[i] = sum ; }
```

Now, torque[ ] is ready to be sent to the ARM D/A and generate forces.

Chapter 10

System Configuration and Hardware Design

Speed, speed, and speed. Real-time response is the major concern of this chapter. I will list some real-time performance data and system delays. These data are useful for designers of real-time simulations.

10.1 Three generations of the ARM system configuration

The ARM system I have used has been evolving for 3 years, and there have been three generations of ARM configurations. A major concern is the update rate. We have used state-of-the-art workstations in different configurations to get the speed.

10.1.1 Masscomp MC500-based system

Delay problems

In this first generation, the delays were significant (tabulated below).

task	cost of time	percentage
Jacobian matrix calculation	3.6 ms	1.2
force and torque calculation	111 ms	35
A/D and D/A	80 msec	25
Ethernet communication and PS300 images	83.7 ms	26.5

Because of the delays listed above, the system could achieve only 3.1 updates per second when everything was included, and the receptor and drug were real molecules of moderate complexity. With simple drugs (a few atoms), the speed was higher, from 5 to 10 updates

per second. This was not good enough to allow any precise measurement of the effectiveness of force display in this application.

10.1.2 SUN3 and SUN4 combined system

Our second configuration used a SUN4 as a compute server, and a SUN3 doing A/D and D/A only. We used the SUN3 for A/D and D/A because the A/D and D/A boards were available only for SUN3 workstations at that time.

The overall update rate was 22 to 30 Hz for molecular docking, even though we used Ethernet communication between a SUN3 and SUN4. However, the effective system delay for force output is doubled, because of the pipelining design.

task	cost of time
Jacobian matrix calculation	1 ms
force and torque calculation	15 ms to 25 ms
A/D (16 channels)	9 ms
D/A (8 channels)	2.4 ms
two-way Ethernet communication (16 channels of integers)	8 ms
SUN4 to PS300 (one command)	5 ms

10.1.3 SUN4-based system

This is the configuration we use now. A SUN4 workstation has replaced the role of the MASSCOMP workstation in the first configuration; that is, A/D, D/A, and computation are all done on one workstation.

The overall update rate is 22 to 30 Hz for molecular docking. In fishing simulation and six-springs simulation, the updating rate is 60 Hz, limited only by the 60 Hz input (synchro transformer). Unlike the second configuration, the system delay is estimated to be the same as the sampling period, and no pipelining in the codes.

task	cost of time
Jacobian matrix calculation	1 ms
force and torque calculation	15 ms to 25 ms
A/D (16 channels)	2.8 ms
D/A (8 channels)	0.8 ms
SUN4 to PS300 (one command)	5 ms

10.2 Mechanical problems

Three mechanical problems are inherent in the E-3 manipulator, which was designed for a maximum force output of 8 pounds. First, the eight-pound maximum force seems too high for delicate perception, so we have to scale the force down. When we do, the friction associated with the cable-driven system is too high. If the synthesized forces are so small that they cannot overcome the static friction, the user does not feel any force. We use a force dial to let the user control the force output scale.

Second, the mechanical linkages to the gear boxes introduce errors due to cable stretching, backlash in the gears, etc. The best precision we can have in our system is 0.5 degrees rotation at each manipulator joint, although the synchro transformer can in principle give 0.1 degree precision. In a molecular world, rotation of a drug in the receptor site by just 0.5 degrees may cause significant force changes. The outside atoms of a drug 10 $\mathring{A}$ in diameter may have moved 0.05 $\mathring{A}$, and the forces vary with high powers of atomic separation.

Third, the manipulator has high mass and therefore cannot be accelerated quickly.

Safety control.

We find the following warnings in the 1976 Grope-II system, where a manipulator is used as a 6-D input and force display [Kilpatrick 76]:

THE MANIPULATOR ARMS ARE INHERENTLY SLOW DEVICES. NO QUICK OR "JERKY" MOTIONS SHOULD BE ATTEMPTED IN USING THEM.

UNLIKE A COMPUTER SYSTEM, THE MANIPULATOR ARMS, IF USED IMPROPERLY, HAVE THE PHYSICAL CAPABILITY OF DESTROYING THEMSELVES AND THEIR ENVIRONMENT. THEREFORE, FOLLOW THE INSTRUCTIONS CAREFULLY AND EXACTLY.

Basically, the same warnings still apply to our 1989 system. To create a user-safe interface, Kilpatrick took two precautions: a dead-man footswitch, on which the user must step to keep power on the servo motors; and safety-range detectors to ensure that the manipulator is always working in a safe region, even when the user releases the handgrip and the manipulator is in free motion.

We provided an external circuit timeout to protect against software collapse. This cuts off the fixed-field current for the servo motors if they do not get updated commands at least every second. We added a door chime that sounds if the ARM's safety-range detectors signal.

10.3 Detailed hardware design in circuit level

The key problems in the hardware design are (circuit diagram plate 1):

1. To get the position and orientation of the handgrip of the manipulator
2. To control the torque output from the six servo motors for six joints in the manipulator

10.3.1 Circuit design to get the position and orientation of the handgrip

The detailed circuit diagrams are listed at the end of this chapter. Let us consider the easier part first. We have a 16-channel analog-to-digital converter, and each channel has 12 bit resolution. The Argonne E-3 manipulator was originally designed to work in a master-slave mode. We modified it so that we have an independent manipulator. There is a synchro transformer associated with each joint of the manipulator, and the output voltage from the transformer is related to the angle of each joint.

The synchro transformer's output is a 60 Hz AC voltage with a maximum peak-to-peak amplitude of 90 volts. More precisely, the output is an amplitude-modulated 60 Hz AC signal, with the amplitude determined by the angle of each joint. In order to get the angle information, we have to know the peak amplitude in run time. The problem becomes how to detect the peak of the signal and measure the amplitude efficiently. Since the output signal from the synchro motor is synchronized to the 60 Hz AC power, the problem of detecting the peak of the signal is greatly simplified. We can detect the peak of the 60 Hz AC power line, and use the peak position thus obtained to sample the synchro transformer output (circuit diagram plate 2).

Our analog-to-digital converter can take a -5 to 5 volt input signal and convert it to digital output. The synchro-transformer output voltages have to be reduced in order to be sampled. We use a voltage divider to attenuate the signal. On the other hand, we found crosstalk problems among the six A/D channels. As we increased the sampling frequency, the crosstalk became increasingly serious. So we tried to match the impedance by using emitter followers (circuit diagram plate 3). The emitter follower is a circuit design that can reproduce the input signal voltage, increase the current gain, and lower the output impedance.

10.3.2 Circuit design to control the torque output from servo motors

We have a 8-channel digital-to-analog (D/A) converter, and each D/A is twelve bit in resolution. Integer output ranging from -2048 to 2047 can be transformed to analog voltages from -5 to 5 volts. The servo motors need 60 Hz AC current both for the field (stator) current and for the control (rotor) current. Moreover, the field current and control current should be 90 degrees out of phase in order to get the maximum torque output. The torque output can be expressed as

```
Torque =  K*I-field*I-control*sin(phase-angle)
```

where K is a constant; I-field and I-control are the field current and control current; phase-angle is the difference of phase angle between the two currents.

Assuming that the servo motor input impedance is a constant in the working range, we can control the torque output by varying the voltage of the control-field signal. With a fixed field of 110 volt 60 Hz AC current, the output force can be as high as 8 pounds if the control field is 110 volts.

Our purpose is to generate a 60 Hz amplitude-modulated signal, with the amplitude determined by the D/A. The following is the circuit design (circuit diagram plate 4). To get the 90-degree phase-shifted signal, we use a 90 degree phase shifter to transform the input AC reference into a phase-shifted signal. The 90-degree phase shifter is nothing but a RC phase-shifting circuit, since we know that the frequency is 60 Hz. The phase-shifted reference is further amplitude-modulated by analog output from the D/A by a multiplier. The amplitude-modulated signal thus obtained cannot be used to drive the servo motor directly, because the voltage is still within -5 to 5 volts. An audio amplifier is used to amplify the signal and to drive the servo motor. The input range for the audio amplifier we got is 300 millivolts, and the output impedance is 70 ohms. Since the servo motor has an input impedance of 300 ohms, we use a transformer to match the impedance. Another attenuator is used to reduce the 5 volts signal to 300 millivolts for audio amplifier input.

10.4 Determine the joint angles of the ARM

Once we obtain the digital readings from the synchro transformer, the next step is to convert these data into joint angles. The software is contained in the procedure **adtoangle.c**.

The conversion to angles is more complicated than we expected. First of all, the synchro-transformer output is not a linear function of the joint angles. In theory, the signal has a

sinusoidal form with a fixed period. We can choose the monotonically increasing part of the function and use a polynomial function to approximate it. A third-order polynomial is enough for it with a precision of 0.5 degrees. It is a pity that one of the angles, the elbow bend, actually was not located within the monotonically increasing range. So the reading was ambiguous as to whether the reading is before or after the peak value. We solved it by opening the gear-box and rotating one of the gears.

To get rid of the 60 Hz AC transformers, we have made a set of potentiometers that can replace the transformers. The actual replacement may happen any time from now on.

Second, we cannot get precise readings from the synchro transformer as a function of angles. The readings are jittering even though the manipulator is fixed in position. In a twelve bit A/D, we observed one to two bit jittering, depending on the joints. In one experiment without using cables, I was able to get a stable reading. I assume the noise is picked up through the circuits and connection cables.

Third, the reading from each channel is not independent of the other! We found that the manipulator was so constructed that it is optimized for manipulation but not for monitoring the angles. For example, if we change the shoulder-bend angle but keep elbow-bend angle fixed, the reading from the elbow-bend joint will change! The reason is that the two gears that control the motion of the elbow and shoulder are not independent, so that the combined torques from the servo motors for the elbow and shoulder are greatly simplified. Although this is meant to simplify the torque output function, the synchro readings are complicated since they share the same pair of gears. Experimental data show that the actual elbow-bend angle equals the unmodified elbow angle minus the shoulder-bend angle, i.e.,

$$Elbow_{new} = Elbow_{raw} - S \tag{10.1}$$

where $Elbow_{new}$ is the actual elbow-bend angle, $Elbow_{raw}$ is the unmodified elbow-bend angle, and S is the shoulder-bend angle.

The other mutually dependent readings come from the wrist-roll and wrist-bend joints. The wrist-roll and bend operation is controlled by a pair of differential gears (Figure 10.1). In order to know the wrist-bend angles, we have to know both readings from the differential gears.

$$Wrist_{bend} = (R+L)/2.0 \tag{10.2}$$

$$Wrist_{roll} = (R+L)/2.0 \tag{10.3}$$

where $Wrist_{bend}$ is the actual wrist-bend angle, $Wrist_{roll}$ is the actual wrist-roll angle, R is the right-gear angle, and L is the left-gear angle.

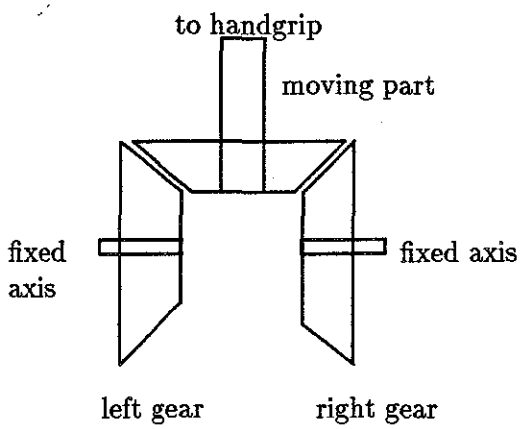
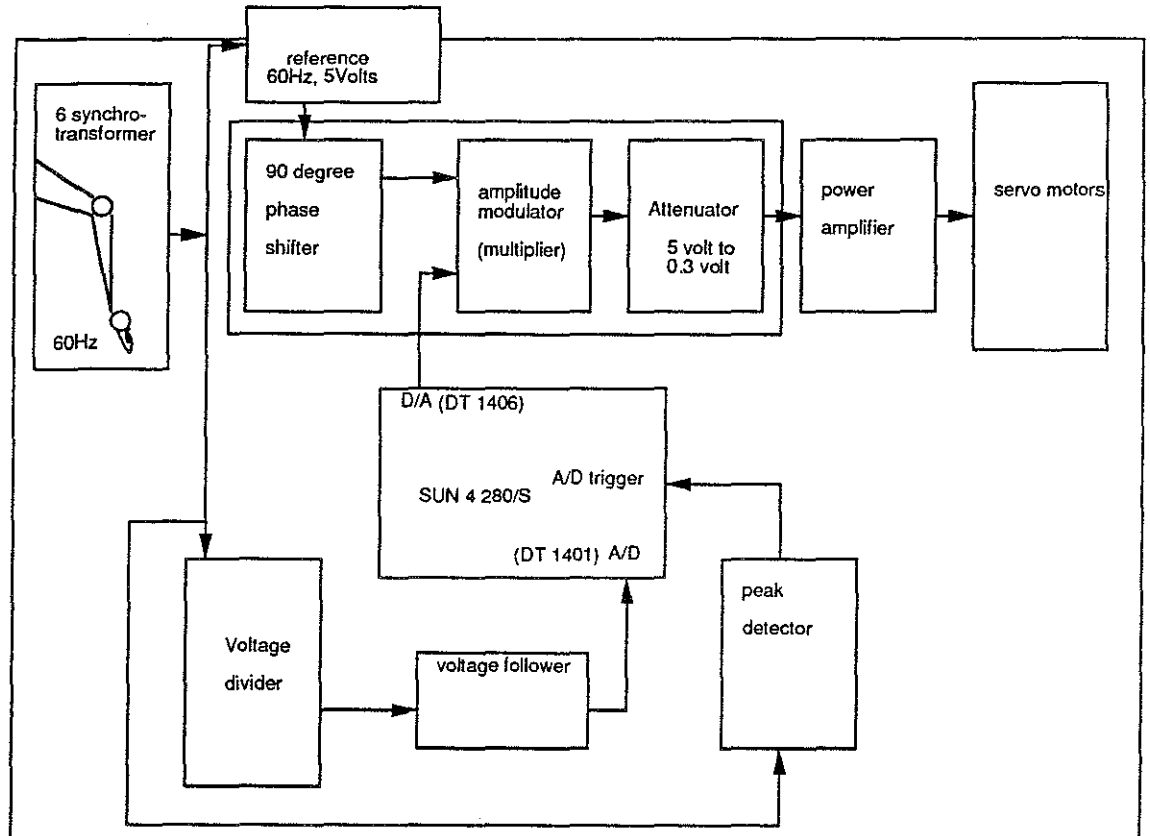


Figure 10.1: Differential gears at the ARM wrist joint.

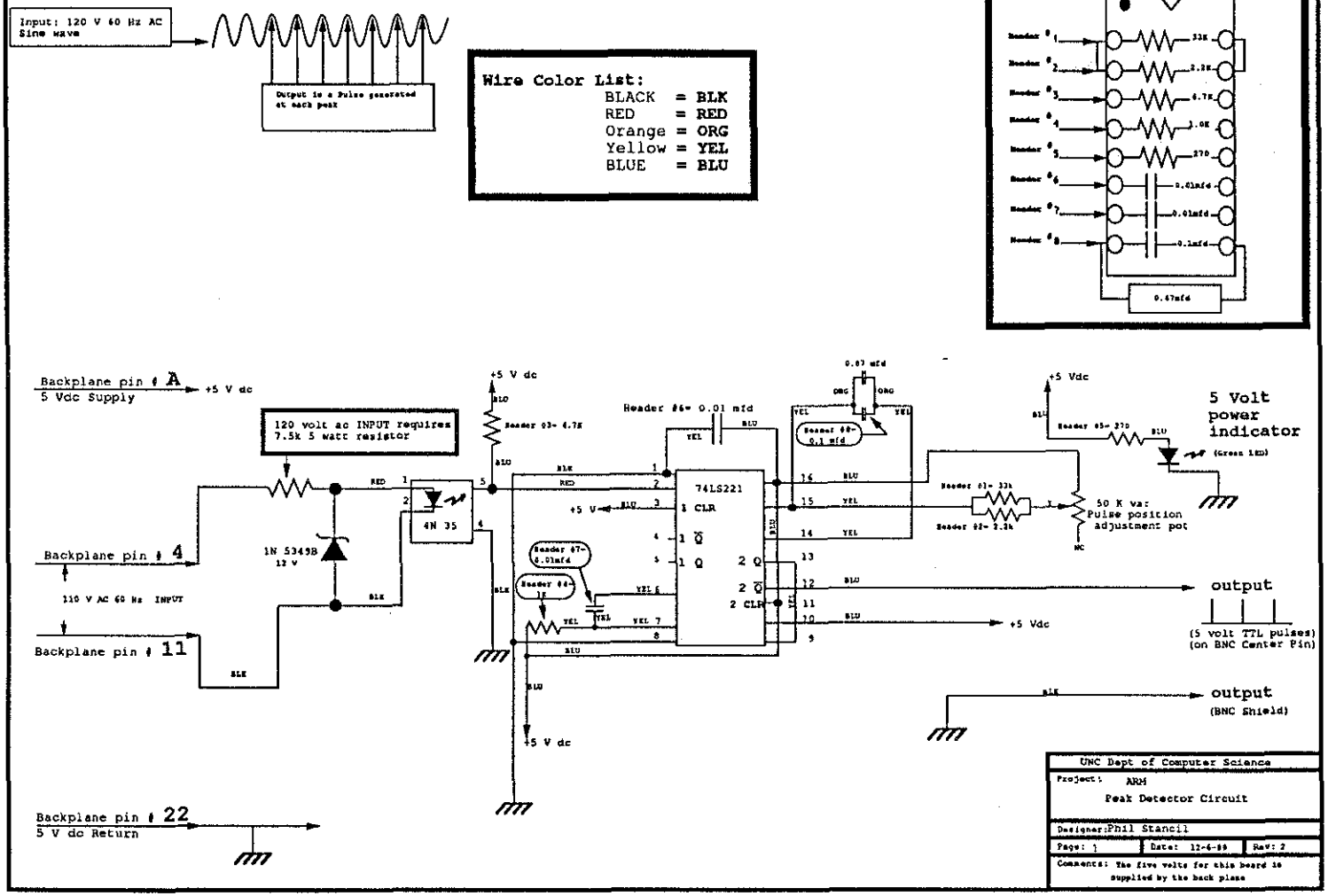
The factor 1/2.0 in Eq. 10.2 means that if both gears rotate in the same direction for $\mathcal{D}$ degrees, the wrist-bend angle is $\mathcal{D}$. On the other hand, if the differential gears rotate in the reverse direction for $\mathcal{D}$ degrees, the resulting wrist-roll angle is two times $\mathcal{D}$, instead of $\mathcal{D}$ only.

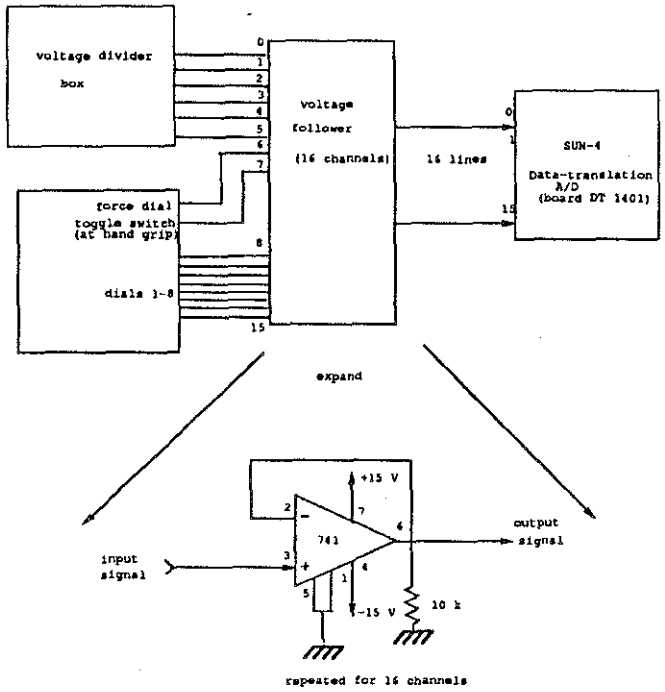


The ARM system block diagrams.

UNC Dept of Computer Science		
Project: ARM		
Designer: Ming Ouh-young		
Page: 1	Date: 4/15/86	Rev:
Comments: ARM controller diagram		

This peak detector generates a pulse each time the 120 Volt 60 Hz input signal reaches a peak.

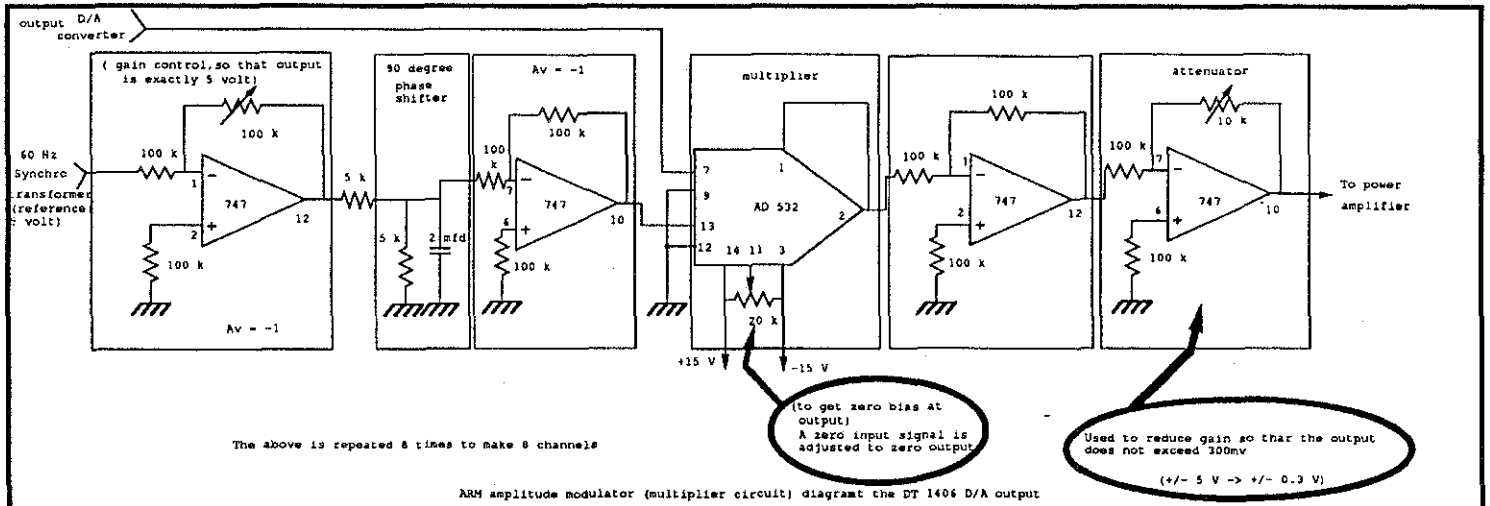




voltage follwer diagram

Comments: To match the impedance from the voltage divider box to the A/D inputs.

UNC Dept of Computer Science		
Project: ARM		
Designer: Ming Ouh-young		
Page: 1	Date: 4/15/88	Rev:
Comments: Voltage follower diagram		



The above is repeated 8 times to make 8 channels

ARM amplitude modulator (multiplier circuit) diagramt the DT 1406 D/A output

UNC Dept of Computer Science		
Project: ARM		
Designer: Ming Ouh-young		
Type: 1	Date: 4/15/86	Rev:
Comments: Amplitude modulator (multiplier circuit)		

Chapter 11

Contribution and Future Work

11.1 Contribution

My thesis work focuses on the generation and presentation of forces and torques, the experimental design and evaluation of force display in real molecular docking problems, and the theory and analysis of creating an illusion of feel.

My contributions to knowledge include

1. Implementation of a real-time 6-D force-feedback computer output system (the GROPE-III) to the human kinesthetic sensing.

2. Analysis of limitations and constraints and corresponding solutions to the generation of forces. I derived several rules that can explain and solve many problems in force-feedback systems.

3. Evaluation of the performance of the 6-D force-feedback device as a tool in docking within a force field consisting of simulated springs. The performance is defined as the task-completion time when the system potential energy is below a certain threshold.

4. One controlled experiment, plus case-by-case study, to evaluate the usefulness of force display in the application of molecular docking. The subjects are biochemists, and the test drug molecules are real research molecules.

5. Development of a new method of docking using visual display of forces and torques for geometric docking, and algorithmic docking for a small search space with more complicated force models, and no kinesthetic display. This method can be applied to workstations.

11.2 Future Work

There are many things to be done in the ARM system, and the following are recommended.

11.2.1 Attack the unknown drug molecules

We should invite biochemists to bring more drug molecules and let them explore the possible solutions. The standard procedures are given in Appendix 2.

11.2.2 Provide a hook to molecular dynamics simulation

The molecular dynamics simulations need a good starting conformation in order to have reliable results. In this sense, the ARM system is efficient in generating good starting positions. The ARM system can generate output files in standard Brookheaven Protein Databank format.

11.2.3 Searching for a desk-top miniARM

The use of a table-top manipulator would make the work space match the screen space, and let the user work on a table, using wrist-finger motions instead of shoulder-elbow-wrist motions. This appears very desirable.

I would anticipate the desk-top miniARM to have smaller viscosity, and thus the system sampling rate would have to be higher. This would be unfortunate. But there is good news. The above analysis assumes that one wants to simulate a spring with one's whole hand to feel it. In order to create an illusion of the force field, the forces presented to one's hand must not be too small in order to feel the difference. In the ARM, we choose a spring constant of 800 N/m in order to let the hand feel it, since the human arm's stiffness can be at most 800 N/m. Now, if only fingers (two or three fingers to hold a miniARM) are used, the molecular force field can be mapped to a smaller spring constant, and a human operator can still feel the differences in spring forces.

Here are the trade-offs. If the effective damping using a miniARM (human fingers + miniARM) is five times as low as that while using the ARM (human arm + ARM), and if the spring constant is reduced by M times, the net effect is equivalent to having a required update rate 5/M times that of the ARM in order to keep the system stable. I estimate the M value at 2.0 to 5.0. In this case, the miniARM system would have to run faster than the ARM.

11.2.4 Combining Turk's collision detection algorithm with 3-D tabulation to control error

The 3-D tabulation method is efficient; however, it is an approximation method. Improved algorithms (in complexity) in molecular mechanics are possible, but we do not see one that works in real time on workstations. Reducing the grid spacing in 3-D tabulation helps to reduce the error, but the memory capacity limits the spacing. The required memory increases as the cube of the number of divisions along one dimension.

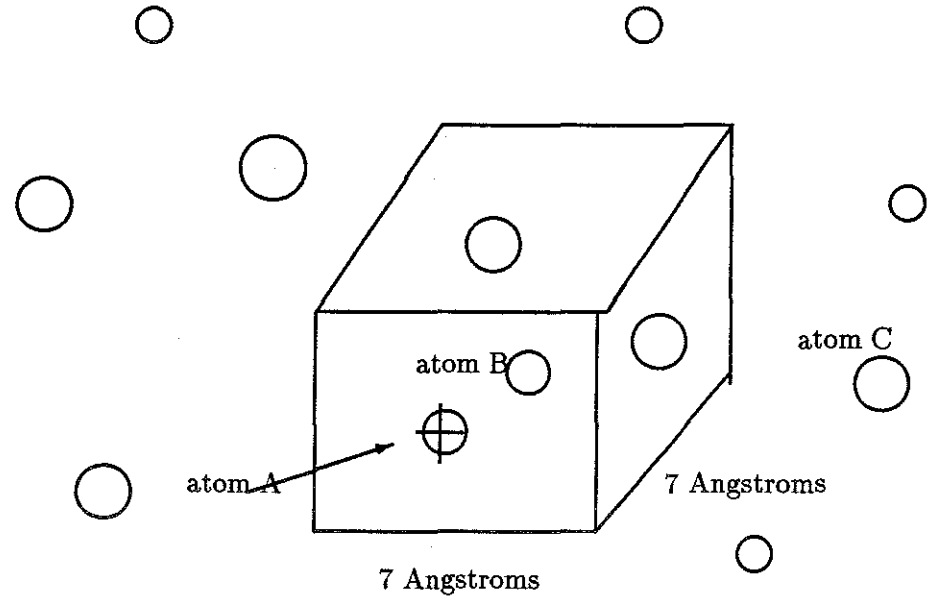
Turk has developed an algorithm that can do collision detection in real time, but it does not calculate forces [Turk 89]. Turk's algorithm creates a sparse 3-D grid, with each grid cell pointing to the nearest atoms within the cube. Because a molecule has a maximum density, the number of atoms being pointed to from a grid is limited.

One possible improvement is to mix the two methods above. Given an atom in an inhibitor, what are the forces exerted on this atom from all the receptor atoms? Let us divide the sources of forces into two classes, the one within a bounding box (with a length of, say, 7 angstroms), and the other outside the bounding box. In theory, if two atoms are more than 3.5 Angstroms apart, Pattabiraman's algorithm can have a maximum error of 3% with 0.5 Angstrom grid spacing.

Now we can create two grids, a dense one for Patabiraman's 3-D tabulation, and a sparse one for Turk's algorithm (Figure 11.1). Turk's algorithm can calculate the forces precisely for atoms that are close (within the bounding box), and Patabiraman's algorithm can calculate forces for atoms that are out of the bounding box, and with an error bound that is tolerable. My preliminary results show that there are a maximum of 7 atoms within a 7x7x7 $\AA$ cubic box for the protein DHFR. Therefore, this mixed algorithm has a complexity of O(n), where n is the number of atoms in a drug molecule. However, the average cost of doing so is to have another constant of around 7. This will be affordable in the near future; for example, a DEC3100 workstation is already three times faster than a SUN4.

11.2.5 A public tool for various simulations in the virtual world

I have never tried to make the ARM a public tool, because the hardware configurations are evolving, the SUN4 that controls the ARM is different from the Pxpl-Planes4 host (a MicroVax), and the control routines are complicated.



Atom A is the target atom, atom B is inside the bounding box, and atom C is outside the bounding box. For atoms outside the bounding box, use Pattabiraman's algorithm, and for atoms inside the bounding box, use Turk's algorithm.

Figure 11.1: A mixed method combining Pattabiraman's and Turk's algorithm.

The ARM configuration is now fixed, and the Ethernet communication between the ARM SUN4 and the Pxpl-Planes4 MicroVax (with a new 3100 CPU) is twice as fast. Three simulations (fishing, six springs, and molecular docking) serve as examples to show how to use the control routines. Therefore, I would expect more users to include the ARM into their virtual world simulations (Russel Taylor and Ronald Azuma, UNC students, have successfully used the ARM in their virtual world projects).

11.2.6 An X.11-window based user interface

The ARM usually runs on a remote workstation that does not have a console at the demo site. For a window-based user interface, this may cause some problems, unless a portable window system is used. X.11 is portable; Sun windows are not.

The user interface should be handled in an event-driven way, and polling should be avoided. The reason is that if the ARM simulation is temporarily suspended because of mouse operations (by polling), the side effect is that the system is out of control. This may cause

disasters. For example, in Minsky's Sandpaper system running on a MacII, one has to use his left hand to hold the joystick while using his right hand to drag the mouse. Otherwise, the joystick may strike back at maximum torque.

11.2.7 High-level functional specifications

It is obvious that force display needs high-level abstraction for some simulations. Low-level mechanical impedance specifications are too tedious for the program design.

For virtual world projects, we should develop a set of routines that accepts abstract input such as *rough, light, oily, springy*, etc., with sliders (knobs) to control them, without worrying about how they are achieved, or if the system will become unstable.

11.2.8 A walkman force-display?

As the head-mounted display (HMD) becomes less expensive and more popular, we would expect an ever increasing demand for force feedback also. A walkman force-display seems to be one of the solutions, because the force display must be always within reach of the user that wears the HMD. I would suggest a walkman force-display that is so small and light-weight that it can be attached to the user's belt, with a hand-control in front of the chest.

A data glove with force feedback at the fingers seems to be another solution. However, the translational and rotational forces are better apprehended through force display. The Cyberspace project [Farmer 89] advocates networked virtual worlds with the HMD, but no specific force feedback. I believe a walkman force-display will supplement the illusions in Cyberspace projects.

Appendix A

The Jacobian Matrix of the ARM

A.1

This appendix gives the complete derivation of the Jacobian matrix for the Argonne E-3 manipulator. Originally the Jacobian matrix describes the differential changes between two coordinate systems, but through the "equivalence of virtual work done in two coordinate systems," the Jacobian matrix can also describe the force and torque transformation between two coordinate systems.

A.1.1 Two useful equations in coordinate, force, and moment transformations

We present questions first, and then give equations as the solutions.

Problem (i). Given a differential translation vector D and rotation vector R in coordinate system E, what is the differential translation and rotation in a new coordinate system T? Solution: If a 4x4 matrix T^* describes the coordinate transformation from frame E to T, and if T^* is expressed as [N, O, A, P], where N, O, A, P are column vectors, then

$$D_x^T = R \cdot (P \times N) + D \cdot N \quad \text{(A.1)}$$

$$D_y^T = R \cdot (P \times O) + D \cdot O \quad \text{(A.2)}$$

$$D_z^T = R \cdot (P \times A) + D \cdot A \quad \text{(A.3)}$$

$$R_x^T = R \cdot N \quad \text{(A.4)}$$

$$R_y^T = R \cdot O \quad \text{(A.5)}$$

$$R_z^T = R \cdot A \quad \text{(A.6)}$$

where $D^T = (D_x^T, D_y^T, D_z^T)$, translation vector in T; $R^T = (R_x^T, R_y^T, R_z^T)$, rotation vector in T; "·" means inner product, and "×" means cross product.

Let $U = [D, R], V = [D^T, R^T]$; the above solution can be expressed as a concise form:

$$V = J * U \tag{A.7}$$

where J is the 6x6 Jacobian matrix.

$$\begin{bmatrix} D_x^T \\ D_y^T \\ D_z^T \\ R_x^T \\ R_y^T \\ R_z^T \end{bmatrix} = \begin{bmatrix} N_x & N_y & N_z & (P\times N)_x & (P\times N)_y & (P\times N)_z \\ O_x & O_y & O_z & (P\times O)_x & (P\times O)_y & (P\times O)_z \\ A_x & A_y & A_z & (P\times A)_x & (P\times A)_y & (P\times A)_z \\ 0 & 0 & 0 & N_x & N_y & N_z \\ 0 & 0 & 0 & O_x & O_y & O_z \\ 0 & 0 & 0 & A_x & A_y & A_z \end{bmatrix} \begin{bmatrix} D_x \\ D_y \\ D_z \\ R_x \\ R_y \\ R_z \end{bmatrix}$$

Problem (ii). Given force F and moment M in coordinate system E, what are the equivalent forces and moments in coordinate system T?

Solution: moments transform in the same manner as differential translations do; forces transform in the same manner as differential rotations do. That is to say, take the equations A.1–A.6 above, replace D by M, R by F, and we have the desired results. If a 4x4 matrix T^* describes the coordinate transformation from frame E to T, and if T^* is expressed as [N, O, A, P], where N, O, A, P are column vectors, then

$$M_x^T = F \cdot (P \times N) + M \cdot N \tag{A.8}$$
$$M_y^T = F \cdot (P \times O) + M \cdot O \tag{A.9}$$
$$M_z^T = F \cdot (P \times A) + M \cdot A \tag{A.10}$$
$$F_x^T = F \cdot N \tag{A.11}$$
$$F_y^T = F \cdot O \tag{A.12}$$
$$F_z^T = F \cdot A \tag{A.13}$$

where the force vector in frame T is $F^T = (F_x^T, F_y^T, F_z^T)$, and the moment in frame T is $M^T = (M_x^T, M_y^T, M_z^T)$; "·" means inner product, and "×" means cross product.

A.1.2 Relating forces and torques at the ARM handgrip to the ARM joint torques

One interesting problem encountered in our system is how to solve the problem of relating forces and moments applied in coordinate frame M to equivalent joint torques and forces.

```
Solution :  joint torques and forces =
            [Transpose of  Jacobian in  M] x [given forces and moments]
Proof:  by the equivalence  of virtual work, i.e.,
```

$$Force * \Delta Displacement = Torque * \Delta Rotation \quad (A.14)$$

$$F^{transpose} * D = T^{transpose} * R \quad (A.15)$$

where force F = [F1, F2, F3, M1, M2, M3], displacement D = [Dx, Dy, Dz, Rx, Ry, Rz], torque T = [T1, T2, T3, T4, T5, T6], rotation R = [R1, R2, R3, R4, R5, R6], and R1 .. R6 are joint angles.

Now, $D = J * R$ (Eq. A.7), J is the Jacobian matrix. So, $F^{transpose}$ * J * R = $T^{transpose}$ * R i.e., $F^{transpose} * J = T^{transpose}$,

$$T = J^{transpose} * F \quad (A.16)$$

end of proof

A.1.3 One way to derive the ARM Jacobian matrix

Suppose our manipulator can be decomposed as A = A1 A2 A3 A4 A5 A6 from the base to the hand, where Ai's are homogeneous coordinate matrices.

To obtain the Jacobian matrix in coordinate frame M, we have to calculate $\partial A/\partial q_i$, where q_i is the angle between joints. The first column in the 6x6 Jacobian matrix can be derived from solving $\partial A/\partial q_1$, where q_1 is the shoulder-bend angle. One does not need to calculate $\partial A/\partial q_1$ directly, because in vector form $D = J * R$, and J can be derived from Eqs. A.1-A.6.

To calculate the second column, just replace A by A2 A3 A4 A5 A6 , and follow Eq. A.13. To calculate the third column, just replace A by A3 A4 A5 A6, and follow Eq. A.13.

Similarly, the fourth to the sixth columns can be obtained. Intuitively, when the upper arm rotates, the handgrip coordinate is related to upper-arm roll, elbow-bend, lower-arm

roll, wrist-bend, wrist-roll, i.e., A2 A3 A4 A5 A6. Similarly, when there is wrist motion only, the handgrip's position and orientation depend on the wrist only, and it is not necessary to know the elbow and shoulder configurations.

A.2 The Jacobian matrix for the Argonne E-3 manipulator

A.2.1 Homogeneous matrix descriptions of the ARM links

Before the Jacobian matrix derivation, I have to describe the meaning of the homogeneous matrix that describes each link in a manipulator. The six joints of the ARM are depicted in Figure. A.1.

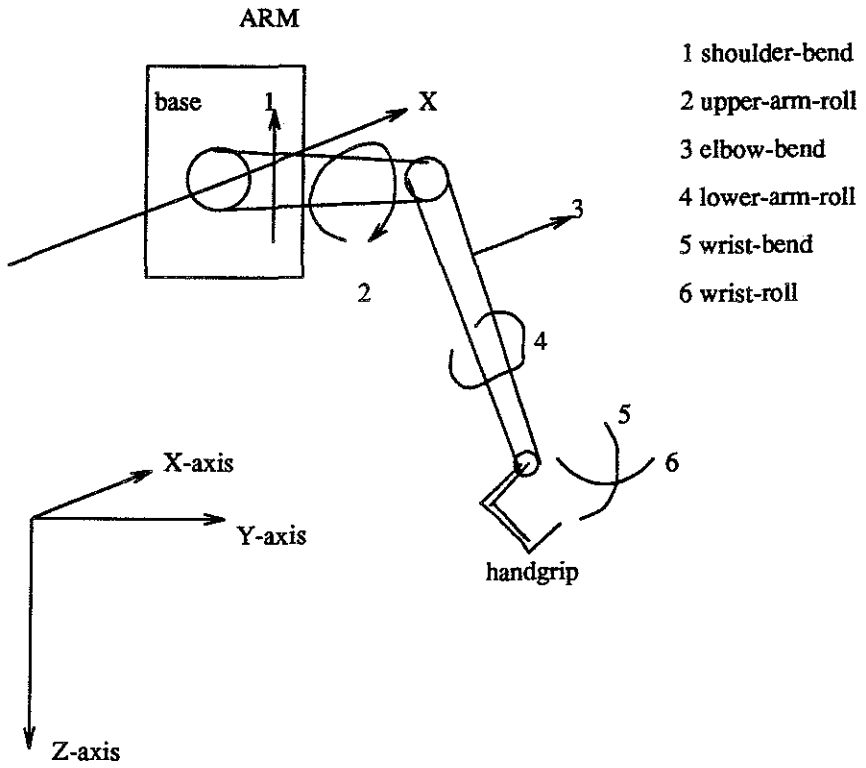


Figure A.1: The six ARM joints.

Shoulder bend: rotating about X axis. Input channel-4 reads the shoulder-bend angle.

```
A1 =  {  1   0   0   0
         0  c4  s4   0
         0 ms4 c4   0
         0   0   0   1 }
```

where "c4" means Cos(shoulder-bend angle), "s4" means Sin(shoulder-bend angle), and ms4 means minus Sin(shoulder-bend angle.

Upper-arm roll : rotating about Y axis. Input channel-1 reads the upper-arm-roll angle.

```
A2 =  { c1   0 ms1   0
         0   1   0  76
        s1   0  c1   0
         0   0   0   1 }
```

where "c1" means Cos(upper-arm-roll angle).

Elbow bend : rotating about X axis. Input channel-7 reads the elbow-bend angle.

```
A3 =  { 1    0    0    0
        0   c7   s7    0
        0  ms7   c7    0
        0    0    0    1 }
```

where "c7" means Cos(elbow bend angle).

Lower arm roll : rotating about Y axis. Input channel-3 reads the lower-arm-roll angle.

```
A4 =  { c3   0  ms3   0
         0   1    0  102
        s3   0   c3   0
         0   0    0   1 }
```

where "c3" means Cos(lower-arm-roll angle), and lower-arm-link length is 102 cm.

Wrist bend : rotating about X axis . Input channel-5 reads the wrist-bend angle.

```
A5 = { 1   0    0   0
       0  c5   s5   0
       0 ms5   c5   0
       0   0    0   1 }
```

where "c5" means Cos(wrist-bend angle).

Wrist roll : rotating about Y axis. Input channel-6 reads the wrist-roll angle.

```
A6 = { c6  0 ms6  0
        0  1   0  0
       s6  0  c6  0
        0  0   0  1  }
```

where "c6" means Cos(wrist-roll angle).

The individual matrix can be constructed directly from a description of the links which is in **argonne.joint** by the program **jointtomat.** When we multiply these matrices together, we use a symbolic matrix multiplier to optimize the number of multiplications. The symbolic matrix multiplier is **symbolmatmul.** The output from the matrix multiplier can be converted to a Fortran or C program directly by **mattof** or **mattoc.** So, in Unix commands, these programs can be chained together by "pipes" such as

```
jointtomat argonne.joint | symbolmatmul | mattof > nest.f
```

A.2.2 Derivation of the Jacobian matrix

First column of the 6 by 6 Jacobian matrix. Let us get the first column of the Jacobian. What is the corresponding differential displacement and rotation of the handgrip in terms of the handgrip coordinate system, if we are given a dq1 rotation in shoulder-link coordinate system about the X axis? Differential displacement D = [0, 0, 0], differential rotation R = [1, 0, 0]*dq1. Hand-grip coordinate frame is given by T = A1*A2*A3*A4*A5*A6 = [N, O, A, P] where N, O, A, P are 1 by 4 column vectors. We can simplify R to be [1, 0, 0] in our calculation, leaving out the scalar factor dq1 for convenience.

Differential displacement D' and Rotation R' can be expressed as

```
Dx' = N . (R x P),  Dy' = O . (R x P),
```

```
Dz' = A . (R x P),
Rx' = N . R,        Ry' = O . R,
Rz' = A . R
where . means inner product and x means cross product.
```

The first column of Jacobian matrix (Eq. A.7) is [Dx',Dy', Dz', Rx',Ry',Rz'], which is

```
[(1)= -out(1,2)*out(4,3)+out(1,3)*out(4,2)
 (2)= -out(2,2)*out(4,3)+out(2,3)*out(4,2)
 (3)= -out(3,2)*out(4,3)+out(3,3)*out(4,2)
 (4)= out(1,1)
 (5)= out(2,1)
 (6)= out(3,1)
]
```

where

```
out(1,1) = ((c1)*(c3)+((ms1)*(c7))*(s3))*(c6)+(((ms1)*(ms7))*(s5)+
           ((c1)*(ms3)+((ms1)*(c7))*(c3))*(c5))*(s6)
out(2,1) = ((ms1)*(ms7))*(c5)+((c1)*(ms3)+((ms1)*(c7))*(c3))*(ms5)
out(3,1) = ((c1)*(c3)+((ms1)*(c7))*(s3))*(ms6)+(((ms1)*(ms7))*(s5)+
           ((c1)*(ms3)+((ms1)*(c7))*(c3))*(c5))*(c6)
out(1,2) = (((s4)*(s1))*(c3)+((c4)*(s7)+((s4)*(c1))*(c7))*(s3))*(c6)+
           (((c4)*(c7)+((s4)*(c1))*(ms7))*(s5)+(((s4)*(s1))*(ms3)+
           ((c4)*(s7)+((s4)*(c1))*(c7))*(c3))*(c5))*(s6)
out(2,2) = ((c4)*(c7)+((s4)*(c1))*(ms7))*(c5)+(((s4)*(s1))*(ms3)+
           ((c4)*(s7)+((s4)*(c1))*(c7))*(c3))*(ms5)
out(3,2) = (((s4)*(s1))*(c3)+((c4)*(s7)+((s4)*(c1))*(c7))*(s3))*(ms6)+
           (((c4)*(c7)+((s4)*(c1))*(ms7))*(s5)+(((s4)*(s1))*(ms3)+
           ((c4)*(s7)+((s4)*(c1))*(c7))*(c3))*(c5))*(c6)
out(4,2) = ((c4)*(c7)+((s4)*(c1))*(ms7))*(102)+(c4)*(76)
out(1,3) = (((c4)*(s1))*(c3)+((ms4)*(s7)+((c4)*(c1))*(c7))*(s3))*(c6)+
           (((ms4)*(c7)+((c4)*(c1))*(ms7))*(s5)+(((c4)*(s1))*(ms3)+
           ((ms4)*(s7)+((c4)*(c1))*(c7))*(c3))*(c5))*(s6)
out(2,3) = ((ms4)*(c7)+((c4)*(c1))*(ms7))*(c5)+(((c4)*(s1))*(ms3)+
           ((ms4)*(s7)+((c4)*(c1))*(c7))*(c3))*(ms5)
out(3,3) = (((c4)*(s1))*(c3)+((ms4)*(s7)+((c4)*(c1))*(c7))*(s3))*(ms6)+
```

```
                (((ms4)*(c7)+((c4)*(c1))*(ms7))*(s5)+(((c4)*(s1))*(ms3)+
                ((ms4)*(s7)+((c4)*(c1))*(c7))*(c3))*(c5))*(c6)
      out(4,3) = ((ms4)*(c7)+((c4)*(c1))*(ms7))*(102)+(ms4)*(76)
```

Second column of the 6 by 6 Jacobian matrix. To get the second column, consider the upper-arm roll. The differential motion can be described as D = [0, 0, 0], R = [0, 1, 0]*dq2, since the upper-arm-roll joint rotates about the Y axis in its coordinate system. So what is the corresponding differential displacement and rotation of the handgrip in terms of the handgrip coordinate system?

Now, T becomes A2*A3*A4*A5*A6 = [N,O,A,P] where N,O,A,P are 1 by 4 column vectors. R can be further simplified to be [0,1,0] , leaving out the factor dq2 for convenience. Differential displacement D' and Rotation R' can be expressed as

```
Dx' = N . (R x P)   Dy' = O . (R x P)
Dz' = A . (R x P)
Rx' = N . R         Ry' = O . R
Rz' = A . R
```

So the second column of Jacobian matrix is [Dx', Dy', Dz', Rx', Ry', Rz'], which is

```
 [ (1)= out(1,1)*out(4,3)-out(1,3)*out(4,1),
   (2)= out(2,1)*out(4,3) - out(2,3)*out(4,1),
   (3)= out(3,1)*out(4,3) -out(3,3)*out(4,1),
   (4)= out(1,2),
   (5)= out(2,2),
   (6)= out(3,2)
  ]
where

    out(1,1) = ((c1)*(c3)+((ms1)*(c7))*(s3))*(c6)+(((ms1)*(ms7))*(s5)+
               ((c1)*(ms3)+ ((ms1)*(c7))*(c3))*(c5))*(s6)
    out(2,1) = ((ms1)*(ms7))*(c5)+((c1)*(ms3)+((ms1)*(c7))*(c3))*(ms5)
    out(3,1) = ((c1)*(c3)+((ms1)*(c7))*(s3))*(ms6)+(((ms1)*(ms7))*(s5)+
               ((c1)*(ms3)+((ms1)*(c7))*(c3))*(c5))*(c6)
    out(4,1) = ((ms1)*(ms7))*(102)
    out(1,2) = ((s7)*(s3))*(c6)+((c7)*(s5)+((s7)*(c3))*(c5))*(s6)
```

```
out(2,2) = (c7)*(c5)+((s7)*(c3))*(ms5)
out(3,2) = ((s7)*(s3))*(ms6)+((c7)*(s5)+((s7)*(c3))*(c5))*(c6)
out(1,3) = ((s1)*(c3)+((c1)*(c7))*(s3))*(c6)+(((c1)*(ms7))*(s5)+
           ((s1)*(ms3)+((c1)*(c7))*(c3))*(c5))*(s6)
out(2,3) = ((c1)*(ms7))*(c5)+((s1)*(ms3)+((c1)*(c7))*(c3))*(ms5)
out(3,3) = ((s1)*(c3)+((c1)*(c7))*(s3))*(ms6)+(((c1)*(ms7))*(s5)+
           ((s1)*(ms3)+((c1)*(c7))*(c3))*(c5))*(c6)
out(4,3) = ((c1)*(ms7))*(102)
```

Third column of the 6 by 6 Jacobian matrix. To get the third column, consider the elbow bend. The differential motion can be described as D = [0, 0, 0], R = [1, 0, 0]*dq3, since the elbow joint rotates about the X axis in its coordinate system. So what is the corresponding differential displacement and rotation of the handgrip in terms of the handgrip coordinate system?

Now, T becomes A3*A4*A5*A6 = [N, O, A, P], where N, O, A, P are 1 by 4 column vectors. Then we get the third column of Jacobian matrix [Dx', Dy', Dz', Rx', Ry', Rz'], which is

```
[ (1)  (s7*s3*c6+c7*s5+s7*c3*c5*s6)*s7*102+
       (c7*s3*c6+ms7*s5+ c7*c3*c5*s6)*c7*102,
  (2)  (c7*c5+s7*c3*ms5)*s7*102+(ms7*c5+c7*c3*ms5)*c7*102,
  (3)  (s7*s3*ms6+c7*s5+s7*c3*c5*c6)*s7*102+
       (c7*s3*ms6+ms7*s5+c7*c3*c5*c6)*c7*102,
  (4)  c3*c6+ms3*c5*s6,
  (5)  ms3*ms5,
  (6)  c3*ms6+ms3*c5*c6.
]
```

Fourth column of the 6 by 6 Jacobian matrix. To get the fourth column, consider the lower-arm roll. The differential motion can be described as D = [0, 0, 0], R = [0, 1, 0]*dq4, since the lower-arm roll joint rotates about the Y axis in its coordinate system. So what are the corresponding differential displacement and rotation of the handgrip in terms of the handgrip coordinate system?

Now, T becomes A4*A5*A6 = [N, O, A, P], where N, O, A, P are 1 by 4 column vectors. Then we get the forth column of Jacobian matrix [Dx', Dy', Dz', Rx', Ry', Rz'], which is [0, 0, 0, s5*s6, c5, s5*c6].

Fifth column of the 6 by 6 Jacobian matrix. To get the fifth column, consider the wrist bend. The differential motion can be described as D = [0, 0, 0], R = [1, 0, 0]*dq5, since the wrist-bend joint rotates about the Y axis in its coordinate system. So what is the corresponding differential displacement and rotation of the handgrip in terms of the handgrip coordinate system?

Now, T becomes A5*A6 = [N, O, A, P], where N, O, A, P are 1 by 4 column vectors. Then we get the fifth coulmn of Jacobian matrix [Dx', Dy', Dz', Rx', Ry', Rz'], which is [0, 0, 0, c6, 0, ms6].

Sixth column of the 6 by 6 Jacobian matrix. To get the sixth column, consider the wrist roll. The differential motion can be described as D = [0, 0, 0], R = [0, 1, 0]*dq6, since the wrist-roll joint rotates about the Y axis in its coordinate system. So what is the corresponding differential displacement and rotation of the handgrip in terms of the handgrip coordinate system?

Now, T becomes A6 = [N, O, A, P], where N, O, A, P are 1 by 4 column vectors. Then we get the sixth coulmn of Jacobian matrix [Dx', Dy', Dz', Rx', Ry', Rz'], which is just [0, 0, 0, 0, 1, 0].

A.3 Special modification at the elbow-bend joint

The Argonne E-3 manipulators are not direct-drive manipulators. In particular cables and aluminum bars are used to control the elbow-joint motion. We have to modify the direct results from the Jacobian matrix, because of the elbow joint of the manipulator. Because there is a parallel bar below the elbow joint to push and pull the forearm, the elbow is no longer a simple revolving joint (Figure A.2). Compensation of torques between the shoulder and elbow motion is required in this case and is given below,

$$shoulder\_bend\ torque^{new} = shoulder\_bend\ torque -$$

$$elbow\_bend\ torque(1 + K * Cos(elbow\_bend\ angle))$$

$$where \quad K = \frac{upper\_arm\ length}{lower\_arm\ length}$$

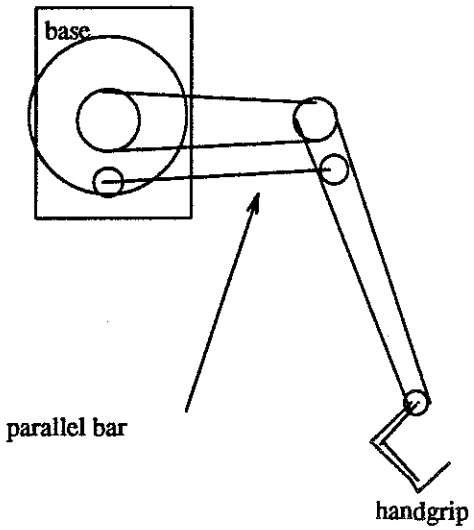


Figure A.2: The special elbow-bend joint with a parallel bar.

Appendix B

Preparing Test Molecules for Molecular Docking

This document describes how to prepare molecules for molecular docking. If a biochemist brings a set of molecules to the laboratory and requests to the working environment to be set up for him, the following procedures are necessary in order to prepare the correct internal files. Notice that since our system tries to use all kinds of visual cues and force cues, the preparation of molecules is not a trivial job.

B.1 Internal files that are necessary for test molecules

What is the necessary information we need for molecular docking using ARM? In general, we need all the information that is in a Brook Heaven Protein Data Bank file, plus the partial charge associated with each atom. The partial charges are important for molecular mechanics force-field calculation. Ask for this information if the biochemist can provide it (usually they can). Actually, all the biochemists that uses AMBER (a commercial package that does local energy-minimization) should have this data. If you already have partial-charge information, skip the next paragraph.

The AMBER package has an internal table that can give partial charges to every atom in a protein according to residue types. The more difficult part is the partial charges for inhibitors. Although there are tables for proteins, nucleic-acids etc., there are no fixed data for general inhibitors. The partial charges can be obtained from GAUSSIAN (a commercial package that does *ab initio* quantum-mechanics calculation), at the cost of extremely high computational complexity and massive storage. What is actually done is to split the inhibitor molecule into several parts, and calculate them independently. The other way is to use tables that consist of similar parts, and use calculated guesses.

The following is the master file format we need.

```
(sample file of the trimethoprim molecule)
f1    f2          f3          f4          f5          f6     f7
-------------------------------------------------------------
N  1.885015   2.609993  -1.040009  -0.280000   1.750000 27
H  2.521528   2.672155  -0.258292   0.340000   1.000000 19
C  2.165012   3.219992  -2.160009   0.840000   1.850000 4
N  3.315006   3.899996  -2.210010  -0.660000   1.750000 27
H  3.584926   4.400570  -3.125142   0.330000   1.000000 19
H  3.977892   3.921083  -1.350805   0.330000   1.000000 19
N  1.415010   3.209987  -3.240005  -0.800000   1.750000 27
C  0.365017   2.259981  -3.280003   0.690000   1.850000 4
N -0.494986   2.279977  -4.359998  -0.600000   1.750000 27
H -1.335397   1.603490  -4.406613   0.310000   1.000000 19
H -0.388265   3.052171  -5.108079   0.310000   1.000000 19
C -0.154979   1.739981  -2.000003  -0.170000   1.850000 4
C  0.695018   1.949990  -0.970006   0.330000   1.850000 4
H  0.500600   1.403194  -0.034746   0.070000   1.375000 21
   where  f1 is the atom name, N for Nitrogen, C for Carbon etc.,
          f2,f2,f3 are the x-y-z  coordinates,
          f5 is the partial charge,
          f6 is the atom radius,
          f7 is the detailed atom type in Peter Kollman's (AMBER)
          classification. This, coordinates, and the partial charge
          define the molecular mechanics force field.
```

The above file can be transformed from the following file, which is available from the AMBER package. For example, the usage is pdbtolg <input-file >output-file.

```
 f1      f2 f3    f4     f5       f6      f7     f8      f9
-------------------------------------------------------------
ATOM      1 N1   U91      1      0.387   2.729  -5.386 -0.622
ATOM      2 HN1  U91      1      0.486   3.305  -6.210  0.432
ATOM      3 C2   U91      1      1.041   3.092  -4.188  1.044
ATOM      4 NA   U91      1      1.837   4.271  -4.192 -1.102
ATOM      5 HNA1 U91      1      2.356   4.552  -3.291  0.521
ATOM      6 HNA2 U91      1      1.941   4.860  -5.098  0.480
```

```
ATOM      7 N3   U91     1       0.936   2.324  -3.013 -0.802
ATOM      8 C4   U91     1       0.221   1.178  -3.011  0.944
ATOM      9 NB   U91     1       0.085   0.385  -1.835 -1.036
ATOM     10 HNB1 U91     1      -0.528  -0.504  -1.839  0.479
ATOM     11 HNB2 U91     1       0.509   0.738  -0.906  0.495
ATOM     12 C5   U91     1      -0.477   0.762  -4.232 -0.225
   where  f2  is the atom sequence,
          f3 is the atom type in AMBER classification, and can
          converted into a integer index in the previous table (f7).
          f4 is the residue name, f5 is the molecule number,
          f6,f7,f8 are the coordinates, f9 is the partial charge.
```

B.2 Preparing protein enzymes

The procedure given above helps prepare the internal file. In addition to that, there is a pre-processing for the molecular force-field calculation.

B.2.1 Energy and force evaluation

Following Pattabiraman's 3-D tabulation approach, we need to pre-calculate the energy tables for all the grid points that cover the protein enzyme. Currently we use 0.5 Angstrom grid spacing, and the whole table for 18x24x28 Angstroms takes 5 Megabytes in memory, if each floating point takes 4 bytes.

The following input file is required to specify the range and grid spacing of the tabulation. To run the energy evaluator, use minggrid < rootname. This takes about 10 to 20 minutes on a SUN4 workstation, depending on the size of the grid. The source file is in /glycine/grip7/arm/grid.c; see the makefile (makegrid).

```
  (file dhfr1.gi, where dhfr1 is the root name, gi is the suffix)
# large molecule file data type
datatype ASCII
# output grid file data type
gridtyp BINARY
#order           XYZ
gridmin -8      -14     -4
#grid points minimum in x,y,z
gridmax  9       9       23
```

```
#grid points maximum in x,y,z
gridres  0.5    0.5     0.5
grid point spacing in x,y,z, here 0.5 Angstrom is used.
termtyp         elect  vdw       short
#calculate three components of the force field
interp  1
#this data is for linear interpolation (interp = 1)  later.
```

B.2.2 PS300 display files

The PS300 has its own display file format. We choose to use the stick model to display a molecule, and the chemical bonds are coded by color— green for Carbon, blue for Nitrogen, red for Oxygen, etc. To generate the display file, for example, use lgto300 < dhfr1.lg (input file) >output-file. The source file lgto300.c is in /glycine/grip7/arm/data.

Once the PS300 display file is generated, there is some header information to be added. The best way is to look at the available working files and copy the header file. For example, the following is a standard file.

```
file: the PS300 display file for DHFR in leucine:/unc/grip/arm/lib/mrh_dhfr.ps,
remove mrh_dhfr from bonds1;
include mrh_dhfr in bonds1;
mrh_dhfr:=begin_s
set color 0,0.000000;
label1 := LABELS  0,   0, 0    ''
      0,   0, 0    ''
      0,   0, 0    ''
      0,   0, 0    ''
      0,   0, 0    ''
;
instance of mrh_dhfr.label1 ;
set color 240,1.00000;
label2 := LABELS  0,   0, 0    ''
;
{vectors of DHFR, coded by color}
set color 240,0.500000;
C0:=vec sep n=716
1.389967, 3.000008, -4.369991
```

```
set color 160,1.000000;

{end of DHFR vectors}

{additional display file reserved for other display purposes, such
as collision detection, hydrogen bonds, etc.}
C4:=vec sep n=200
-5,  -5,  -2
-5,  -5,  -2
-5,  -5,  -2
-5,  -5,  -2
;
instance of mrh_dhfr.C4;
instance of mrh_dhfr.C5;
instance of mrh_dhfr.C6;
{end of display file}
```

The above ASCII file can be converted into binary file, so that it takes less time to down-load the display file. This is done by using psout -d <ascii-display-file >binary-display-file.

B.3 Preparing inhibitor molecules with rotatable bonds

B.3.1 Determination of the display tree

In Figure B.1, there are 14 possible rotatable bonds. If we twist any one of them, the result is a change in molecular conformation. The way we build the PS300 display file is according to a display tree (see Figure B.2 for example).

There are two technical concerns in implementing the display tree. First, we do not want to update all the transformation matrices on the PS300: that would take too much time in communication between a SUN4 and a PS300. What we want is to send the matrix that is associated with the rotatable bond currently being manipulated, and nothing else. Second, the addition of more than ten 4 by 4 matrices will slow down the PS300 display time. Therefore, we should be able to pick up a subset of possible rotatable bonds in run time, and concentrate on that. Furthermore, once we pick up a new root for a molecule in run time, the PS300 display file should be reconfigured and sent to PS300.

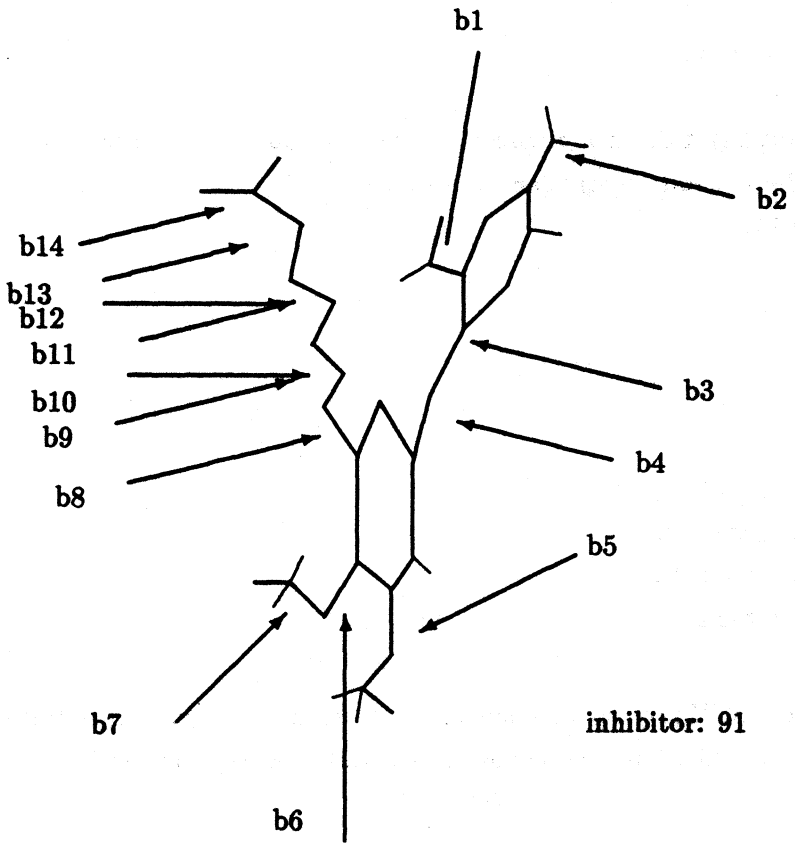
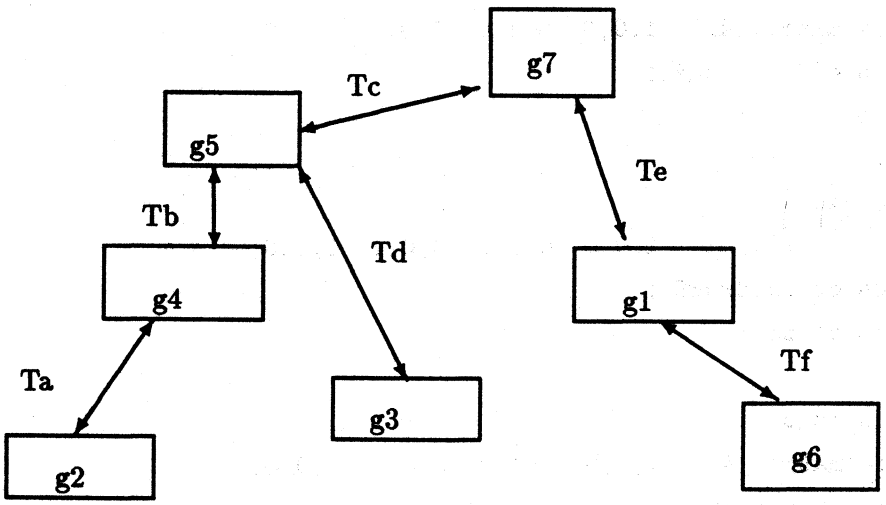


Figure B.1: Inhibitor 91 with possible rotatable bonds.

B.3.2 PS300 display files and files of hierarchy information

The following is a sample PS300 file for the inhibitor 91. The inhibitor 91 has 9 rotatable bonds, and each rotatable bond is controlled by a 4x4 matrix (matR1, matR2, etc.). Once the PS300 vectors are generated for different groups of the inhibitor, the following header file is necessary.

```
remove mrh_trm from bonds2;
include mrh_trm in bonds2;
mrh_trm := begin_s
        instance of s0;
        instance of s1;
        end_s ;
        s1 := begin_s
```



where g1, g2, .., g7 are subgroups of atoms,

Ta, Tb,.., Tf are 4 by 4 matrices that describe the rotational degree.

If g4 is picked up as the root, the transformation from g4 to g6 is Tb*Tc*Te*Tf, and all 4 by 4 matrices are calculated in run time.

Figure B.2: A display tree with transformation matrices.

```
matR1 := matrix_4x3  1,0,0  0,1,0  0,0,1  0,0,0;
instance of rotdrug1 ;
instance of s2 ;
 end_s ;
s3 := begin_s
matR3 := matrix_4x3  1,0,0  0,1,0  0,0,1  0,0,0;
instance of rotdrug3 ;
instance of label1;
instance of s4 ;
end_s;
s2 := begin_s
matR2 := matrix_4x3  1,0,0  0,1,0  0,0,1  0,0,0;
instance of rotdrug2 ;
instance of s3 ;
 end_s ;
```

```
        s4 := begin_s
        matR4 := matrix_4x3  1,0,0  0,1,0  0,0,1  0,0,0;
        instance of rotdrug4 ;
        instance of s5 ;
        end_s ;
        s5 := begin_s
        matR5 := matrix_4x3  1,0,0  0,1,0  0,0,1  0,0,0;
        instance of rotdrug5 ;
        instance of s6 ;
        end_s ;
        s6 := begin_s
        matR6 := matrix_4x3  1,0,0  0,1,0  0,0,1  0,0,0;
        instance of rotdrug6 ;
        instance of s7 ;
        end_s ;
        s7 := begin_s
        matR7 := matrix_4x3  1,0,0  0,1,0  0,0,1  0,0,0;
        instance of rotdrug7 ;
        instance of s8 ;
        end_s ;

        s8 := begin_s
        matR8 := matrix_4x3  1,0,0  0,1,0  0,0,1  0,0,0;
        instance of rotdrug8 ;
        instance of s9 ;
        end_s ;
        s9 := begin_s
        matR9 := matrix_4x3  1,0,0  0,1,0  0,0,1  0,0,0;
        instance of rotdrug9 ;
        end_s ;
{Actual display vectors begin here}
s0 :=begin_s
{end of vectors}
```

B.4 Preparing the solvent accessible surfaces for molecules

There is a useful representation of the active site: the solvent accessible surfaces [Connolly, 80]. Because it takes a few minutes in SUN4, it cannot be done even interactively. However, since the protein enzyme is a fixed structure in our system, the surfaces can be generated by pre-processing.

B.4.1 How to generate the Connolly surfaces for PS300 and Pixl-Planes4?

The surface-generating program is **ms.c**, and it needs several input files. For the usage, please refer to "MS User's Manual" (by Michael Connolly). The source programs are in /glycine/grip3/arm/data/connolly.

The above program can generate vectors that are not color coded. We have to calculate the charge potential on these vectors and color-code them. In general, the neutral potential is represented by green (similar to noncharged Carbon), the positively charged potential is blue (similar to hydrogens attached to Nitrogen), and the negatively charged potential is red (similar to nagatively charged Oxygen). The source code is /glycine/grip3/arm/data/connolly/potentialps.c. Please look at the documentation to run it. An example is potentialps; default input files are conn.vu, which contains the surfaces generated by **ms**, and standard.lg, which can be any of the master protein file, and output file is standard output.

B.4.2 How to display partial surfaces that are relevant to docking?

We do not want to display all the surfaces surrounding a protein enzyme, because our focus is on the active site. Therefore, it is necessary to specify the atoms that are around the active site, and let other atoms be invisible for surface calculation. The simplest way is to draw a sphere around the center of the known active site, and include all the atoms inside the sphere as important ones in determining the surfaces. This usually produce very good partial surfaces. Please refer to "TRB User's Manual," and "PPMS User's Manual" (written by Michael Connolly).

I am not able to generate partial surfaces in polygons, even though all the vector points including surface normals are there. This means that I am not able to put correct surfaces on Pxpl-Planes-4, because my results either have holes in it, or the tiling is wrong. This problem is related to the tiling problem in 3-D: given a set of points and surface normals, find a proper tiling. It is still an open question.

Appendix C

Control Routines for the ARM

C.1 ARM driver and control routines

The following routines are important in order to use the arm. The name in parentheses after the procedure header is the name of the file where the routine is found.

```
------------------------------------------------------------
A list of routine names
   armopen(): open A/D and D/A
   armclose(): close A/D and D/A
   armread1(): read 16 channels
   armwrite1(): write 8 channels
   nest_(): Jacobian matrix calculation
------------------------------------------------------------


armopen(), armclose()
(/grip7/arm/sys/dt/sun4dt/dtio.c)
These routines open and close the arm device driver, and they must
be called before (and then after) any routines that access the arm.
armopen() returns -1 if there is an error opening the arm.
Note that a force output of (0,0,0,0,0,0) should be sent to
armwrite1() before the arm is closed.  This will stop force generation by
the arm after it has been closed.  If this is not done, then the arm will
continue to generate the last force it was told to even after it has been
closed.
```

```
armread1(data)
int data[] ;
(/grip7/arm/sys/dt/sun4dt/dtio.c)
This routine will read the voltage amplitude from each of the arm
rotation sensors, and from each of the dials placed on the lower arm.  This
routine will wait until it receives a synchronization signal from the
external clock before sampling.  This signal occurs at a rate of 60 Hz.
As a result, multiple calls to this routine will each act as a 1/60 second
pause in the program.  The first call may, of course, take from zero time
to 1/60 second depending on the synchronization with the pulse.
The integers returned are from -2048 through 2048, and represent
the amplitude on channels 0-15 of the Analog to Digital converter that
is in glycine.
The numbers returned represent voltages from the following arm
sensors (NOTE THAT THIS DOES NOT CORRESPOND TO CHANGES IN JOINT ANGLE IN
A LINEAR FASHION.  USE adtoangle TO TURN THESE VALUES INTO ANGLES).
This routine will return the value -1 if there an error is found
while reading the values.
data[0] = a switch, view mode swicth (should not be scaled)
data[1] = Upper arm roll.  Rot Y, counterclockwise
data[2] = a switch, the clutch  (should not be scaled)
data[3] = Lower arm roll.  Rot Y, counterclockwise
data[4] = Shoulder bend.  Rot X, counterclockwise
data[5] = Wrist bend, Rot X, counterclockwise **
data[6] = Wrist roll, Rot Y, counterclockwise **
data[7] = Elbow bend.  Rot X, counterclockwise
data[8] = Y-axix rotation
data[9] = dial 1
data[10]= dial 2
data[11]= dial 3
data[12]= dial 4
data[13]= dial 5
data[14]= dial 6
data[15]= a switch, lock/unlock
** data 5 and 6 are differential angles
```

```
armwrite1(data)
int data[] ;
(/grip7/arm/sys/dt/sun4dt/dtio.c)
This routine sends force output to the joint motors in the arm.
There are eight channels of force output, fed through the Digital to
Analog converter that is in glycine.  The number sent to each is from
-2048 to 2047, and represents the percent of maximum torque to deliver
to that joint.  -2048 represents the maximum negative torque and 2047
represents the maximum positive torque.  The actual torque sent to that
joint depends on the maximum torque, which is set via dials on the
various amplifiers between glycine and the arm controls.  Note that the
arm has six degrees of freedom, and a seventh for the pinchers if they
are in place.
This routine returns -1 if there an error is encountered while
writing the values.
data[0] = not used
data[1] = shoulder rotate (upper arm roll)
data[2] = elbow bend
data[3] = lower arm rotate
data[4] = shoulder bend
data[5] = wrist gear A
data[6] = wrist gear B
data[7] = not used


adtoangle( data, angle )
int  data[];
float  angle[];
(/grip7/arm/sys/dt/sun4dt/adtoangle.c)
      main function :   convert from the raw AD ( Masscomp ) readings
                        into  joint angles in Radians.
      input parameter :  AD raw readings.
                         (old raw reading are remembered)
      output parameters :  joint angles in radians.
      polynomial approximation, 3rd degree.
      i.e. angle = a* X*X*X + b* X*X + c*X +d .
```

Coefficients a, b, c, and d are obtained from experimental data.
If input raw data do not change by a significant amount,
then the old data and old angle are used for noise
thresholding.

To keep the output angles stable if there is only a
small amount of movement, the old data and the new
data are compared. In some cases, the old angle and
the input angle are compared in order to decide the
thresholds.

```
nest_(jangs,out,force,fout,jaco,fpm,ss7)
float jangs[8]; /* The INPUT vector of joint angles */
float out[4][4]; /* The OUTPUT transform matrix T6 */
float jaco[6][6]; /* The OUTPUT Jacobian matrix */
float force[6]; /* Force and moment AT ORIGIN */
float fout[6]; /* Above force as seen at hand */
float fpm[3]; /* Value of f x p + m */
float ss7[2]; /* sine and cosine of elbow joint OUT*/
(/grip7/arm/fast/nestja.f)
This is a Fortran subroutine that computes the Jacobian, as well as
the output force given the joint coordinates and the desired force. This
routine can be called from a C program, and the order of array storage is
matched to that of a normal C routine.
Note that the name of the Fortran routine is 'nest', not 'nest_'.
The underbar following is added to tell the linker that this routine was
not compiled using the c compiler.
force[0] = Fx
force[1] = Fy
force[2] = Fz
force[3] = Mx
force[4] = My
force[5] = Mz
```

The force fout must be multiplied by the transpose of jaco to turn

```
it into joint torques: F = fout x jaco

F[0] = maps to data[4] (shoulder bend)
F[1] = maps to data[1] (shoulder rotate)
F[2] = maps to data[2] (elbow bend)
F[3] = maps to data[3] (lower arm rotate)
F[4] = \__ map to the differential gears data[5] and data[6]
F[5] = /

Note that these are the ideal desired joint torques.  However, the
arm was designed with gear ratios differing for the different joints on the
arm.  As a result, these ideal joint torques must be scaled by 1/ratio to
convert them into the torque that will be generated by the motors.
Notice that  data[1]
and data[2] must be scaled by 1/102, and data[4] by 1/72 to convert them into
proper force outputs.
The differential gears map to the motor torques in the following way:
data[5] = sqrt(2)/2.0 * (F[4]+F[5]);
data[6] = sqrt(2)/2.0 * (F[4]-F[5]);
```

Appendix D

Running an ARM Demo

Welcome to the ARM molecular docking system. We have three demo programs using the ARM. (1) A fishing simulation. In this simulation, one holds the fishing pole and pulls in the fish, while the fish tries to escape. One can use dial number one to reel in the fishing line (by turning the dial clockwise), and can also release the fishing line (by turning the dial counter-clockwise). When the fishing-line tension is too much, it will break, and so the user gets a penalty of 30-feet loose fishing line. (2) A simulation of six-springs. One attempts to find the potential energy minimum in a 6-D space defined by six Hooke's Law springs attached to a manipulable object. The goal is to find a zero-force position.

(3) Molecular docking. The system uses an ARM to simulate the interaction forces between two molecules. This is a new approach for analytic drug (medicine) design. The big molecule is a protein receptor, called DHFR. The small drug molecule is designed to be either an anti-bacterial drug, or an anti-cancer drug; in our demo, it is called trimethoprim, a useful medicine for middle ear infections and other bacterial diseases. The ARM system has been used to find the binding positions for several drug analogues within 20 minutes, while the same solution cannot be found by a systematic search algorithm (together with AMBER molecular mechanics modeling program) within one week on an IBM 3081 mainframe computer in RTP.

Please follow the instructions below

Power-up section

(0) While running the demo, press capital 'Q' key on the SUN terminal to stop it. You need at least 65% of the CPU power of a SUN-4 in order to run the ARM demo fast enough to create a smooth motion. There is a way to make the ARM demo run in higher priority by typing hipri process-number (be sure you are within the **hipri** group) where the process number of the ARM demo can be found by

```
ps -aux|more.
```

(1) Open two windows on a sun working station (Threonine, the sun3 that is in the ARM room), one for demo programs, the other for instructions and explanation of the demos.

Remote login to the machine glycine by rlogin glycine (for both windows).

(2) Change your directory in both windows by cd /glycine/grip7/arm/fast.

(3) In one window, read the instructions by vi README (or view README) and follow the instructions carefully. (If you got a printed user manual, it is a copy of the README file.)

(4) If you face the ARM, there is a big screen (about 2.5 by 3 feet) in front of you. Below the screen, on the lower-left side, there is a set of 4 power switches, labeled as **main power** etc. Turn on all the 4 switches.

(5) Find the handgrip of the ARM, and move it to the height of your chest, where you can easily hold it. If the ARM stays in other "out of range" positions, you will hear a continuous alarm (door bell) from below the big screen.

(6) On your right hand there is a book shelf. On the fourth level from the top on the book shelf, there is a white box with a label on the lower right part reading **shutter controller**. Turn the power switch on. This is the stereo-plate controller, and it lets you see stereo images later.

(7) If you face the ARM, there is another book shelf to your left. On the fourth level, there are several black plastic eye glasses. Take them and hand them out to all the visitors. They need the stereo glasses to see stereo images.

(8) On the right-hand side of the big screen there is a PS300 picture system (with a 19 inch small screen), and there is a power switch on the lower-right side below the PS300 screen. Turn on the power switch, and you will hear the noise of fans. There is a set of 8 dials in front of the PS300 screen; when you start the demo programs later, there will be some LED lights for each dial. There is another PS300 keyboard, and you will push some keys on it later.

(9) On the ground in front of the big screen, there is an ARM foot switch that looks like a 4 inch square grey flat box. This is the safety switch; unless you turn it ON by stepping on it there will be no forces from the ARM. You will see a red light turned ON in the upper front part of the ARM.

There is a light-control dial on the wall to the left of the big screen. Dim the room lights by turning the knob to a reading of 5; this will enhance the image quality on PS300.

There are two demo versions here: the first demo, **demo_string** simulates a virtual bar suspended in space by six springs. Each spring has a different spring constant. The purpose of this small demo is to get familiar with the functions of the ARM. The user should move the ARM in such a way that finally there is no force or torque from the ARM, and so the six springs balance each other.

Running the demo program section

(10) Run the first demo program by executing **demo_string** in another window. It takes 30 seconds to load the images for the PS300. Now, you should see three spheres on the PS300 screen, and there are messages from the SUN 3 windows.

Find a dial on the PS300 dial box labeled **scale**, and turn it freely until you can see the whole image with three spheres. Next, you will see a vertical "energy thermometer" on the PS300 screen, and you can adjust its position anywhere on the screen by using the lower-row dials labeled **tran(slate) X** and **tran(slate) Y**. Try moving the two dials a little bit, and you will see the corresponding changes.

Please wear the stereo eye glasses now. Adjust the PS300 dial labeled **stereo** until the images look stereo to you.

(11) Hold the ARM hand grip. There is a trigger in the hand grip. Press the trigger and rotate the ARM anywhere you like. The image will move as you move. The trigger is like

a button on a mouse, and releasing the trigger is equivalent to releasing the button on a mouse; nothing happens when you release the trigger.

(12) On top of the handgrip there is another toggle switch. Press it once with you thumb, and move the ARM again. You will see that only the central object moves while the others stay. Press the switch again and move the ARM; you are back to the original mode. Only when you are in the mode where the central object moves can you feel the forces from the ARM. In another mode, the viewing mode, a small golden word **VIEW** will appear on the lower-right corner of the PS300 screen.

In front of the handle (handgrip) there is a knob (a dial) on the aluminum plate. This is the "force sensitivity dial," which controls the sensitivity of the force output. Turning it clockwise increases the sensitivity. Turn it clockwise to the halfway point. Let the dial stay at this position all the time.

(13) Have fun now by moving the ARM to a position where you do not feel any forces and torques.

(14) Press an 'h' (HELP) key on the SUN 3 keyboard, and you will see a help manual. Press 'h' several times and you will see more. Press the 'G' (Geometry) key, and there will be six additional golden lines on the PS300 screen, these are the current positions of the six simulated springs. If you press the ARM trigger, a golden line will appear on the energy bar, showing the minimum global system potential energy. Tt is your goal to reach there by moving the ARM later.

Press the 'N' (No geometry) key, and, if you hold the ARM trigger again, they will disappear.

(15) In general, a visitor may need 60 seconds to reach a minimum-energy position from a random starting position. If you would like to try another trial with different springs, press 'J' (increment test set) on the SUN 3 keyboard.

This is the end of the first demo. If you want to stop the demo program, press the 'Q'(quit) key.

Starting the second demo, demo_dhfr

(16) Now we are going to give the real molecular docking demo. Type "demo_dhfr" in the demo window. It takes almost one minute to load the program and its images. Be patient until you see ***** Welcome to the docking system! ****** in the demo window.

(17) Repeat the second half of step (10) above in order to get the proper scale, stereo, and energy bars.

(18) On the top row of the PS300 keyboard, there are many function keys, each of them with a label, showing the corresponding images associated with them. Press the 'Shift Line-local' (two keys at the same time) keys on the PS300 keyboard to enable the keyboard to accept commands.

Press "surf 1," and you will see a dotted image appear. Press the "surf 1" key again, and the surface will disappear, just like a toggle switch. It is the receptor site surface of the protein molecule. Try "bonds 1," press it several times, and you will see the results.

During this demo, you have to turn it ON or OFF several times in order to give the visitors a good perception. Each time you turn "surf 1" ON, please turn "bonds 1" OFF.

"Surf 1" controls the image of the dotted surface of a protein; "bonds 1" controls the image of the stick model (bonds only) of the protein. Usually, visitors like "surf 1"; however, sometimes they like "bonds 1".

(19) Here we have an awkward implementation. When you press any keys on the PS300 keyboard, the SUN host does not know it! Therefore, each time you turn "surf 1" ON, you also have to press a 'M' (molecular surface) key for the SUN 3, in order for it to acknowledge the SUN host, and show the yellow flashing bump vectors on PS300 when you move the drug molecule. You cannot see the flashing vectors unless you move the drug molecule and feel strong bump forces.

Similarly, when you turn "bond 1" ON, you have to press the 'B' (bond model) key, doing similar task.

(20) Push the trigger on the ARM handgrip and push the "mode switch" on top of the handle with your thumb. Move the handgrip freely and try pushing the trigger and mode

switch several times until you know their functions. Toggle the "mode switch" until only the drug molecule moves (object mode).

Move the ARM handgrip freely, and you will feel the interactive forces between the two molecules. Enjoy it. The yellow flashing vector tells you that two atoms are too close to each other. Your goal is to find a position where you do not feel strong forces, and do not see the flashing bump vector. This is a possible binding position for two molecules, and you are going to find more local energy minima if you have time.

(21) On top of the ARM handgrip, there are four dials attached to the lower arm. Rotate the dials labeled 1 to 3 slowly while pressing the trigger; you will see the drug molecule deform itself by a rotatable bond. Each dial here controls one rotatable bond in the drug molecule.

Play with the dial and see if you can find other bonding positions while changing the shape of the drug molecule.

To show labels on the protein enzyme: You can show the residue names of the protein enzyme (DHFR) by typing '&' on the SUN3, and '7' to turn it OFF. This is interesting to most chemists.

(22) The energy thermometer tells you the current binding energy of the two molecules. If the energy falls below zero, it is a good sign, meaning that the two molecules attract each other. This is exactly what you need. The best minimum energy is around -50 Kcal/mol in the energy bar. See if you can find it. Good luck.

Starting the third demo: fishing simulation

I have implemented two versions of the fishing simulation— one on Pixel-Planes4, the other one on PS300. For the demo on the PS300, type demo_fish. For the demo on Pixel-Planes4, you need two windows. One window for Pixel-Planes4, one for Glycine. The reason is that I use a UNIX socket between the two machines. In the Pixel-Planes4 window, type

```
cd /glycine/grip3/arm/fast/fish_pxpl
fishload
```

In the glycine window, type demo_fishpxpl. In the Pixel-Planes4 window, there will be two entries for data. The first one asks for the remote machine name; please type in **glycine.**

The second one asks for the processID; please type in the processID shown on the Glycine window.

Now you have to change the viewing point from a joystick-box, which is in front of the Pxpl-Planes-4 monitor. Do not choose an arbitrary viewpoint. Just make the room, the fish, and the fishing pole appear and look big enough.

There is a circular "cake" similar to a score board floating in the air. One of the cake pieces will turn red if there is tension in the fishing line. Another piece will turn green, if you can position the fish inside a blue box on the upper-left corner. Use dial number one (with a red label: 1) to reel-in or release the fishing line. Enjoy the fishing; the fish you catch weighs about one pound, and her name is Wanda.

END OF DEMO, POWER DOWN section

(23) This is the end of the last demo. Press the capital 'Q' key one the SUN3 to quit the demo program.

(24) Turn OFF (a) the PS300 power switch (see step 8, if you forgot its position), (b) the shutter controller box power switch (step 6), (c) the ARM power switch.

(25) THE END.

Bibliography

[Amber 80] This is a widely used molecular mechanics/dynamics program, and was developed by Peter Kollman in UCSF, CA. Distributed by NIEHS, Research Triangle Park, NC 27709.

[Baker 81] D. J. Baker, C.R. Beddell, J.N. Champness, P.J. Goodford, E.E.A. Norrington, D.R. Smith and D.K. Stammers, "The Binding of Trimethoprim to Bacterial Dihydrofolate Reductase," *FEBS LETTERS*, pp 49-52, Vol 126, No 1, April 1981.

[Batter 72] Batter, J. J. and Brooks, F. P., Jr. GROPE-1: "A Computer Display to the Sense of Feel," Proc. IFIP 1972, 759-763.

[Beddel 84] C. R. Beddel, "Designing Drugs to Fit a Macromolecular Receptor," *Chem. Soc. Rev.*, No. 13, 279-319, 1984.

[Bejczy 76] A. K. Bejczy, "Performance Evaluation of Computer Aided Manipulator Control," Proceedings, *IEEE International Conference on Cybernetics and Society*, Washington D.C., November 1976.

[Bejczy 80] A. K. Bejczy, "Sensors, Controls, and Man-machine Interface For Advanced Teleoperation," *Science*, 208, 1327-1335, 1980.

[Bier 86] E. A. Bier, "Skitters and Jacks: Interactive 3D Positioning Tools," *Workshop on Interactive 3D Graphics*, Proc. 183-196, Chapel Hill, October, 1986.

[Billeter 87a] M. Billeter, T.F. Havel, I.D. Kuntz, "A New Approach to the Problem of Docking Two molecules: The Ellipsoid Algorithm," *Biopolymers*, Vol. 26, 777-793, 1987.

[Billeter 87b] M. Billeter, T.F. Havel, and K. Wuthrich, "The Ellipsoid Algorithm as a Method for the Determination of Polypeptide Conformations from Experimental Distance Constraints and Energy Minimization," *Journal of Computational Chemistry*, Vol. 8, No. 2, 132-141, 1987.

[Billeter 88] M. Billeter, A.E. Howard, I.D. Huntz, and P. A. Kollman, "A New Technique To Calculate Low-Energy Conformations of Cyclic Molecules Utilizing the Ellipsoid Algorithm and Molecular Dynamics: Application to 18-Crown-6," *J. Am. Chem. Soc.* 1988, 110, 8385-8391.

[Brooks 77] Frederick P. Brooks, Jr., "The Computer Scientists as Toolsmith: Studies in Interactive Computer Graphics." In information 77. B. Gilchrist, ed. North-Holland Pub. Co., Amsterdam. pp. 625-634, 1977.

[Brooks 88] Frederick P. Brooks, Jr., "Grasping Reality Through Illusion: Intercative Graphics Serving Science." Invited keynote address at the Fifth Conf. on Computers and Human Interaction, Washington, D.C., May 17, 1988. Published in CHI'88 Proceedings, May 1988, 1-11. Addison wesley, 1988.

[Brunger 88] A. T. Brunger, "Crystallographic Refinement by Simulated Annealing on Supercomputers," *Cray Channels*, 16-19, Fall 1988.

[Capowski 71] Capowski, J. J., "Remote Manipulators as a Computer Input Device." M.S. Thesis, University of North Carolina, Chapel Hill, North Carolina, 1971.

[Chen 88] M. Chen, J. Mountford, A. Sellen, "A Study in Interactive 3-D Rotation Using 2-D Control Devices," *ACM SIGGRAPH88*, Proc., 121-129, Vol. 22, No.4, August, Atlanta, 1988.

[Cherubino 88] Catherine Cherubino, "Tactile Simulation: Exploration and Application." BS thesis, Massachusetts Institute of Technology, May 1988.

[Colgate 89] Ed Colgate, and Neville Hogan, "An Analysis of Contact Instability in Terms of Passive Physical Equivalents," *Proceedings of IEEE Robotics and Automation Conference 89*, pp. 404-409, Scottsdale, Arizona, 1989.

[Connolly 83] Connolly, Michael L., "Solvent-Accessible Surfaces of Proteins and Nucleic Acids," *Science*, Vol. 221, No. 4612, 709-713, August 1983.

[Dean 87] P. M. Dean, "Molecular foundations of drug-receptor interactions." Cambridge University Press, 1987. Chapter 3.

[Dearfield 88] David Dearfield, personal communication, 1988.

[Dixon 80] L. C. Dixon, E. Spedicato, G.P. Szego, "Nonlinear Optimization: Theories and Algorithms." Birkhauser Boston, 1980. 429-471.

[Ecker 83] J.G. Ecker, "An Ellipsoid Algorithm For Nonlinear Programming," *Mathematical Programming*, 27, 83-106, 1983.

[Farmer 89] F. Randall Farmer, "Cyberspace: Getting There From Here." *Journal of Computer Game Design*, pp. 4-6, 1989.

[Feldman 86] Richard Feldman of NIH. Joystrings was first implemented by him.

[Fitts 54] P. M. Fitts, "The Information Capacity of The Human Motor System In Controlling The Amplitude of Movement," *Journal of Experimental Psychology*, 381-391, Vol. 47, No. 6, June, 1954.

[Foley 87] James D. Foley, "Interfaces for Advanced Computing," *Scientific American*, Oct. 1987, 126-135.

[FRODO] FRODO is a program, written by T. Alwyn Jones, which allows protein construction from an experimental electron density map.

[GAUSSIAN 85] This program uses molecular orbital methods for the determination of structural and spectroscopic properties, as well as thermodynamic stabilities and energies of molecular interactions. It was developed by Warren Hebre, University of California at Irvine, CA, and distributed by Scott D. Kahn, University of Illinois, Urbana.

[Goertz 54] R.C. Goertz, and R.C. Thompson, "Electronically Controlled Manipulator," *Nucleonics*, pp46-47, 1954.

[Goertz 61] Goertz, R. C. et al, "The ANL Model 3 Master Slave Manipulators – Its Design and Use in a Cave," *Proc. of the Ninth Conference on Hot Laboratories and Equipment*, Washington, D.C., United States Atomic Energy Commission, 1961.

[Greengard 88] L. Greengard and V. Rokhlin, "Rapid Evaluation of Potential Fields in Particle Systems," MIT Press, Cambridge, 1988.

[Hall 84] D. Hall and N. Pavitt, "An Appraisal of Molecular Force Fields for the Representation of Polypeptides," *Journal of Computational Chemistry*, Vol. 5, No. 5, 441-450, 1984.

[Hannaford 88] Blake Hannford, and Robert Anderson, "Experimental and Simulation Studies of Hard Contact in Force Reflecting Teleoperation," *IEEE International Conference on Robotics and Automation*, Proceedings, pp 584-589, Philadelphia, April 1988.

[Hannaford 89] Blake Hannaford, "A Design Framework for Teleoperators with Kinesthetic Feedback," *IEEE Transactions on Robotics and Automation*, Vol. 5, NO. 4, August 1989.

[Hayward 88] Vincent Hayward and Antal Bejczy, Workshop On Shared Autonomous And Teleoperated Manipulator Control, *IEEE International Conference on Robotics and Automation*, Philadelphia, April 1988.

[Hogan 87] Neville Hogan, "Stable Execution of Contact Tasks Using Impedance Control," *Proceedings of IEEE Robotics and Automation Conference 87*, pp 1047-1054, Raleigh, NC, 1987.

[Hogan 89] Neville Hogan, "Controlling Impedance at the Man/Machine Interface," *Proceedings of IEEE Robotics and Automation Conference 89*, pp. 1626-1631, Scottsdale, Arizona, 1989.

[Ishikawa 89] Hiroshi Ishikawa, Chihiro Sawada, Kei Kawase, "Stable Compliance Control and Its Implementation for a 6 D.O.F. Manipulator," *Proceedings of IEEE Robotics and Automation Conference 89*, pp. 98-103, Vol.1, Scottsdale, Arizona, 1989.

[Jacquot 81] Raymond G. Jacquot, "Modern Digital Control Systems." Marcel Dekker Inc., New York, pp 51-53, 1981.

[Janssen 85] P. A. J. Janssen, "Strategies in pharmaceutical Research," *Endeavour*, No. 9, 28-33, 1985.

[Karfunkel 86] H. R. Karfunkel, "A Fast Algorithm for the Interactive Docking Maneuver with Flexible Macromolecules and Probes," *Journal of Computational Chemistry*, Vol. 7, No. 2, 113-128, 1986.

[Kazerooni 89] H. Kazerooni, "Human/Robot Interaction via the Transfer of Power and Information Signals," *Proceedings of IEEE Robotics and Automation Conference 89*, pp. 1632-1647, Scottsdale, Arizona, 1989.

[Kilpatrick 76] Paul Jerome Kilpatrick, *The Use of Kinesthetic Supplement in an Interactive System*, Ph.D dissertation, Computer Science Department, University of North Carolina at Chapel Hill, 1976.

[Kim 87] Won S. Kim, Frank Tendick, Stephen R. Ellis, and Larence W. Stark, "A Comparison of Position and Rate Control for Telemanipulations with Consideration of Manipulator System Dynamics," *IEEE Journal of Robotics and Automation*, Vol. RA-3, No. 5, pp 426-436, October 1987.

[Kirkpatrick 83] S. Kirkpatrick, C.D. Gelatt, Jr., M.P. Vecchi, "Optimization by Simulated Annealing." *Science*, pp. 671-680, Volume 220, No. 4598, 1983.

[Kuntz 82] I.D. Kuntz, J. M. Blaney, S.J. Oatley, R. Langridge, and T.E. Ferrin, "A Geometric Approach to Macromolecule-ligand Interactions," *Journal of Molecular Biology*, 161, 269-288, 1982.

[Kuyper 82] Lee Kuyper, "Receptor-based Design of Dihydrofolate Reductase Inhibitors: Comparison of Crystallographically Determined Enzyme Binding with Enzyme Affinity in a Series of Carboxy-Substituted Trimethoprim Analogues." *J. Med. Chem.*, pp 1120-1122, No. 25, 1982.

[Kuyper 89] Lee Kuyper, private communication.

[Lanman 80] Lanman, J. M., *Movement and the Mechanical Properties of the Intact Human Elbow Joint*, Ph.D dissertation, Department of Psychology, M.I.T., 1980.

[Lederman 79] Susan J. Lederman, "Auditory Perception," *Perception*, Vol. 8, pp 93-103, 1979.

[Lederman 88] S. J. Lederman, and R.A. Browse, "The Physiology and Psychophysics of Touch," NATO ASI Series. Vol. F43. Sensors and Sensory Systems for Advanced Robotics, Edited by P. Dario, Springer-Verlag Berlin Heidelberg 1988.

[Li 82] R. Li and C. Hansch, "A comparison by QSAR, Crystallography, and Computer Graphics of the Inhibition of Various Dihydrofolate Reductases by 5-(X-Benzyl)-2,4-diamino-pyrimidines," *Quantitative Structure - Activity Relationships in pharmacology and Biology*, 1-2, 1,7, 1982.

[Lipscomb 87] James Lipscomb, 1987. private communication.

[Matthews 77] D. A. Matthews, R. A. Alden, J.T. Bolin, S.T. Freer, "Dihydrofolate Reductase: X-ray Structure of the Binary Complex with Methotrexate," *Science* 197, 452-455, 1977.

[McCormick 87] B. H. McCormick, T. A. Defanti, and M. D. Brown, eds., "Visualization in Scientific Computing," *Computer Graphics*, Vol. 21, No. 6, Nov. 1987.

[Minsky 84] M. R. Minsky, "Manipulating Simulated Objects with Real-World Gestures Using a Force and Position Sensitive Screen," *ACM Computer Graphics*, Vol. 18, No. 3, July 1984.

[Minsky 90] Margaret Minsky, Ming Ouh-young, Oliver Steele, Frederick P. Brooks, Jr., Max Behensky, "Feeling and Seeing: Issues in Force Display," to appear in *Proc. of 1990 Symposium on Interactive 3D graphics*, Snowbird, Utah, March 1990.

[Moore 88] M. Moore, J. Wilhelms, "Collision Detection and Response for Computer Animation," *ACM SIGGRAPH88*, Proc., 289-298, Vol. 22, No.4, August, Atlanta, 1988.

[Murray 88] Murray, W.R., *Essential Factors in Modeling the Modulation of Impedance about the Human Elbow.* Ph.D dissertation, Department of Mechanical Engineering, M.I.T, 1988.

[Oatley 84] S. Oatley, J. Blaney, R. Langridge and P. Kollman, "Hormone-Protein Interactions," *Biopolymers*, 23, 2931, 1984.

[O'Donnel 87] T. J. O'Donnel and K. D. Mitchell, "3D Docking Device for Molecular Modeling," *Molecular Graphics*, Vol. 5, No. 2, June 1987.

[Ouh-young 88] Ming Ouh-young, Michael Pique, John Hughes, Neela Srinivasan, Frederick P. Brooks, Jr. "Using a Manipulator for Force Display in Molecular Docking," *IEEE Robotics and Automation Conference*, Proceedings, Vol 3, pp 1824-1829, Philadelphia, April 1988.

[Ouh-young 89] M. Ouh-young, D.V. Beard, F.P. Brooks, "Force Display Preforms Better than Visual Display in a Simple 6-D Docking Task," Proc. *IEEE International Conference on Robotics and Automation*, pp 1462-1466, Arizona, May 1989.

[Palmer 87] T. C. Palmer, *Docktool: A Dynamically Interactive Raster Graphics Visualization for the Molecular Docking Problem.* M.S. thesis, UNC-CH, 1987.

[Paul 81] Richard Paul, *Robot Manipulators: Mathematics, Programming, and Control*, The MIT Press, 1981.

[Pattabiraman 85] Nagarajan Pattabiraman *et al*, "Computer Graphics and Drug Design: Real Time Docking, Energy Calculation and Minimization," *Journal of Computational Chemistry*, Vol. 6, p432-436,1985.

[Pottle 80] C. Pottle, R. Tuttle, R. Kinch, and H. Scheraga, "Conformational Analysis of Proteins: Algorithms and Data Structures for Array Processing," *Journal of Computational Chemistry*, Vol. 1, No. 1, 46-58, 1980.

[Russo 88] Massimo A. Russo, "A Control Analysis of A Hybrid Motor-Brake System For Tactile Simulations." BS thesis, Massachusetts, May 1988.

[Smith 88] M. J. Smith, "Tactile Interface for Three-Dimensional Computer-Simulated Environments: Experimentation and the Design of a Brake-Motor Device," M.S. Thesis, Mechanical Engineering Dept., Massachusetts Institute of Technology, 1988.

[Steelman 68] Steelman, H. S., *The GROPE-I System: An Analysis of Friction and Blacklash Problems.* M.S. Thesis, University of North Carolina, Chapel Hill.

[Suguna 87] K. Suguna *et al*, "Binding of a Reduced Peptide Inhibitor to the Aspartic Proteinase from Rhizopus Chinensis: Implications for a Mechanism of Action," *Proc. Natl. Acad. Sci. USA.* Vol. 84, pp. 7009-7013, October 1987, Biochemistry.

[Sutherland 65] Sutherland, I. E. "The Ultimate Display," *IFIP 65 (International Federation of Information Processing '65)*, Proceedings, pp 506-508, 1965.

[Stark 88] Lawrence W. Stark, Won S. Kim, Frank Tendick, "Cooperative Control In Telerobotics," *IEEE International Conference on Robotics and Automation*, Philadelphia, April 1988.

[Stewart 89] Kent Stewart, private communication.

[Turk 89] Greg Turk, "Interactive Collision Detection for Molecular Graphics," 1989, MS thesis, Computer Science Department, University of North Carolina at Chapel Hill.

[Weiner 84] S. J. Weiner, P. A. Kollman, D. A. Case, U. C. Singh, C. Ghio, G. Alagona, S. Profeta, Jr., and P. Weiner, "A New Force Field for Molecular Mechanical Simulation of Nucleic Acids and Proteins," *J.A.Chem.Soc.* 1984, 106, 765-784.

[Wlassich 86] John J. Wlassich, *Nonlinear Force Feedback Impedance Control*, MS thesis, Massachusetts Institute of Technology, February 1986.