



University of Pennsylvania  
**ScholarlyCommons**

---

Technical Reports (CIS)

Department of Computer & Information Science

---

January 1989

## Software Tools for Dynamic and Kinematic Modeling of Human Emotion

Ernest M. Otani  
*University of Pennsylvania*

Follow this and additional works at: [https://repository.upenn.edu/cis\\_reports](https://repository.upenn.edu/cis_reports)

---

### Recommended Citation

Ernest M. Otani, "Software Tools for Dynamic and Kinematic Modeling of Human Emotion", . January 1989.

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-89-43.

This paper is posted at ScholarlyCommons. [https://repository.upenn.edu/cis\\_reports/705](https://repository.upenn.edu/cis_reports/705)  
For more information, please contact [repository@pobox.upenn.edu](mailto:repository@pobox.upenn.edu).

---

## Software Tools for Dynamic and Kinematic Modeling of Human Emotion

### Abstract

Human body modeling has been undertaken in both the fields of biomechanics and computer graphics. Historically, each approach has lacked some of the advantages of the the other. This project further develops one model used for human task studies and computer animation by improving motion realism and facilitating user interaction with the model. Realism is provided by an interface that links a general purpose mechanism simulator with the JACK graphics environment and a prototype human figure with realistic mass and joint properties based on studies in the biomechanics literature. Improved interaction is achieved through software tools which can position several of the figures joints simultaneously. Also, a tool is developed for calculating the mass and inertia properties of an arbitrary polyhedron based on its geometry and an assumption of constant density. Finally, suggestions are offered for future study.

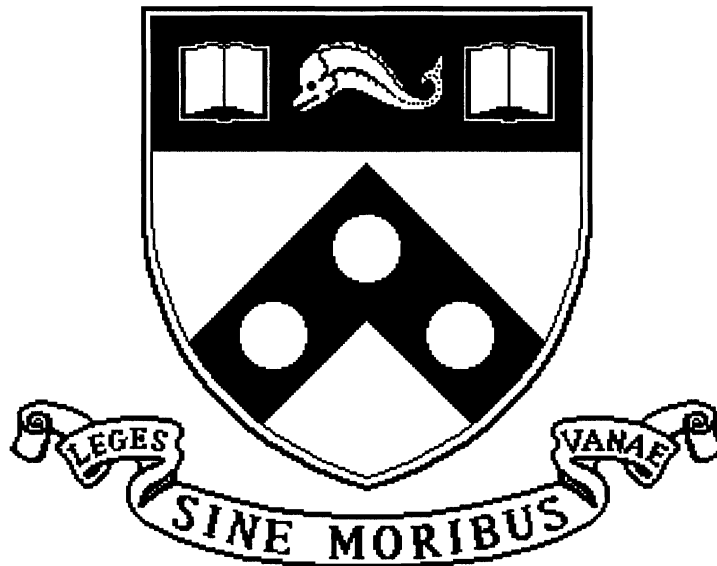
### Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-89-43.

**Software Tools For  
Dynamic And Kinematic  
Modeling Of Human Emotion**

**MS-CIS-89-43  
GRAPHICS LAB 28**

**Ernest M. Otani**



**University of Pennsylvania  
School of Engineering and Applied Science  
Computer and Information Science Department  
Philadelphia, PA 19104-6389**

**1989**

**Software Tools For  
Dynamic And Kinematic  
Modeling Of Human Motion**

**MS-CIS-89-43  
GRAPHICS LAB 28**

**Ernest M. Otani**

**Department of Computer and Information Science  
School of Engineering and Applied Science  
University of Pennsylvania  
Philadelphia, PA 19104-6389**

**July 1989**

**Acknowledgements:**

**This research is partially supported by Lockheed  
Engineering and Management Services, Pacific  
Northwest Laboratories B-U0072-A-N, the  
Pennsylvania Benjamin Franklin Partnership, NASA  
grants NAG-2-426, and NGT-50063, NSF grants  
MCS-8219196-CER, IST-86-12984, IRI84-10413-A02  
and DMC85-16114, and ARO grants  
DAAG29-84-K-0061, DAA29-84-9-0027 including  
participation by the U.S. Army Human Engineering  
Lab.**

UNIVERSITY OF PENNSYLVANIA  
SCHOOL OF ENGINEERING AND APPLIED SCIENCE

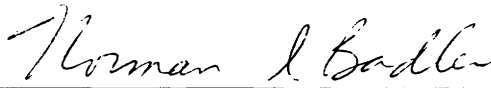
**SOFTWARE TOOLS  
FOR  
DYNAMIC AND KINEMATIC  
MODELING OF HUMAN MOTION**

Ernest M. Otani

Philadelphia, Pennsylvania

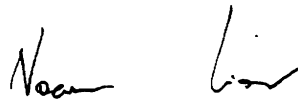
August, 1989

A thesis presented to the Faculty of Engineering and Applied Science of the University of Pennsylvania in partial fulfillment of the requirements for the degree of Master of Science in Engineering for graduate work in Mechanical Engineering and Applied Mechanics.



---

Norman I. Badler  
Advisor



---

Noam Lior  
Graduate Group Chair

## **Abstract**

Human body modeling has been undertaken in both the fields of biomechanics and computer graphics. Historically, each approach has lacked some of the advantages of the the other. This project further develops one model used for human task studies and computer animation by improving motion realism and facilitating user interaction with the model. Realism is provided by an interface that links a general purpose mechanism simulator with the JACK graphics environment and a prototype human figure with realistic mass and joint properties based on studies in the biomechanics literature. Improved interaction is achieved through software tools which can position several of the figures joints simultaneously. Also, a tool is developed for calculating the mass and inertia properties of an arbitrary polyhedron based on its geometry and an assumption of constant density. Finally, suggestions are offered for future study.

To Dr. Theodore Toshiro Otani and Tomie Kojima Otani for the learning, Elaine  
Miye Otani for the enthusiasm, and Dr. Niels Fujio Otani for the curiosity.

## Acknowledgements

I wish to express my gratitude to my advisor Dr. Norman Badler whose encouragement and enthusiasm allowed me to explore a field new to me. Without his input and support this thesis would not have been possible.

I would also like to thank my many colleagues in the University of Pennsylvania Computer Graphics Research Lab who nurtured the development of my project with sound advice, sharp criticism and many excellent ideas. In particular: Cary Phillips for his help and collaboration in developing software for Jack, and Phil Lee for helping me get started, teaching me about DYSPAM and steering me clear of many wrong turns.

This research is partially supported by Lockheed Engineering and Management Services, Pacific Northwest Laboratories B-U0072-A-N, the Pennsylvania Benjamin Franklin Partnership, NSF Grants IST-86-12984 and DMC85-16114, NASA Grants NAG-2-426 and NGT-50063, NSF CER Grant MCS-82-19196, and ARO Grant DAAG29-84-K-0061 including participation by the U.S. Army Human Engineering Laboratory.

Thanks also to Evan Strassberg, for the many insightful conversations about DYSPAM, and my father, Dr. Theodore Otani, for answering my many anatomy questions and helping me find references.

Finally, I would like to thank Kevin Donovan, Rich Quach and Wendy Wagner for their friendship and support.



# Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduction</b>                                      | <b>1</b>  |
| <b>2</b> | <b>Previous Work</b>                                     | <b>5</b>  |
| 2.1      | Crash Simulators and Ejection Seat Studies . . . . .     | 5         |
| 2.2      | Other Body Property Studies . . . . .                    | 8         |
| 2.2.1    | Gait Studies . . . . .                                   | 8         |
| 2.2.2    | Non-Impact Joint Studies . . . . .                       | 8         |
| 2.3      | The Graphics Environment . . . . .                       | 11        |
| 2.3.1    | Animation Models . . . . .                               | 11        |
| 2.3.2    | Graphics Lab Software . . . . .                          | 12        |
| <b>3</b> | <b>Mover: A Dynamics Driver for the Jack Environment</b> | <b>14</b> |
| 3.1      | Data Representations . . . . .                           | 14        |
| 3.1.1    | DYSPAM files . . . . .                                   | 15        |
| 3.1.2    | Peabody figure files . . . . .                           | 17        |
| 3.1.3    | Joint Transformation Conventions . . . . .               | 20        |
| 3.2      | Ernest: Converting the Input . . . . .                   | 21        |
| 3.2.1    | Converting Revolute Joints . . . . .                     | 21        |
| 3.2.2    | Converting Spherical Joints . . . . .                    | 21        |
| 3.3      | Pebble: Converting the Output . . . . .                  | 24        |
| 3.4      | Integrating the Programs . . . . .                       | 25        |
| <b>4</b> | <b>The Body Model</b>                                    | <b>28</b> |
| 4.1      | Modeling Joint Properties . . . . .                      | 28        |
| 4.1.1    | Active and Passive Joint Properties . . . . .            | 29        |
| 4.1.2    | Selecting a Stiffness Function . . . . .                 | 30        |
| 4.1.3    | Stiffness Properties of Spherical Joints . . . . .       | 31        |
| 4.1.4    | Stiffness Properties of Revolute Joints . . . . .        | 32        |
| 4.1.5    | Non-linear Springs: Data Representation . . . . .        | 33        |
| 4.2      | The Prototype Simulation Figure . . . . .                | 33        |
| 4.2.1    | Elbow Springs . . . . .                                  | 35        |
| 4.2.2    | Knee Springs . . . . .                                   | 35        |
| 4.2.3    | Shoulder Springs . . . . .                               | 36        |
| 4.2.4    | Hip Springs . . . . .                                    | 39        |

|          |  |           |
|----------|--|-----------|
| 4.2.5    | Waist Springs . . . . .                          | 42        |
| <b>5</b> | <b>Positioning Tools</b>                         | <b>44</b> |
| 5.1      | Locating the Center of Mass . . . . .            | 44        |
| 5.2      | Testing for Figure Stability . . . . .           | 45        |
| 5.3      | Generating Mass and Inertia Data . . . . .       | 46        |
| 5.4      | Parametric Shoulder Positioning . . . . .        | 55        |
| 5.5      | Multiple Interactive Joint Positioning . . . . . | 59        |
| <b>6</b> | <b>Conclusion</b>                                | <b>63</b> |
|          | <b>References</b>                                | <b>65</b> |
| <b>A</b> | <b>Figure Definition for the Prototype Body</b>  | <b>69</b> |

# List of Figures

|      |   |    |
|------|---|----|
| 3.1  | DYSPAM Input File: structure . . . . .  | 15 |
| 3.2  | DYSPAM Input File: ndyspamfile . . . . .  | 16 |
| 3.3  | DYSPAM Input File: jackforce . . . . .  | 17 |
| 3.4  | Peabody Figure File . . . . .   | 18 |
| 3.5  | Zifyjoint: Algorithm for Handling Restriction on Revolute Joints . . .  | 22 |
| 3.6  | Pebble: Converting DYSPAM output to Jack input . . . . .  | 26 |
| 3.7  | Information Flow of Mover . . . . .   | 27 |
| 4.1  | Data format for specifying non-linear springs . . . . .   | 33 |
| 4.2  | Elbow moment-displacement relationship: Engin's elbow data and prototype elbow model . . . . .  | 35 |
| 4.3  | Knee moment-displacement relationship: CAL3D data and prototype knee model . . . . .  | 36 |
| 4.4  | Shoulder moment-displacement relationship for x axis: Engin subject data and prototype shoulder model . . . . .   | 37 |
| 4.5  | Shoulder moment-displacement relationship for y axis: Engin subject data and prototype shoulder model . . . . .   | 38 |
| 4.6  | Shoulder moment-displacement relationship for z axis: Engin subject data and prototype shoulder model . . . . .   | 38 |
| 4.7  | Illustration of the directional dependence of the passive hip moment. $M_{HFE}$ is the moment for leg motion in the direction of increasing extension and $M_{HEF}$ is the moment for increasing flexion. . . . . | 39 |
| 4.8  | Hip moment-displacement relationship for flexion: Yoon and Mansour subject data and prototype hip model. . . . .  | 40 |
| 4.9  | Prototype hip moment-displacement relationship for ab/adduction. . . . .  | 41 |
| 4.10 | Prototype hip moment-displacement relationship for medial/lateral rotation. . . . .   | 41 |
| 4.11 | Prototype waist moment-displacement relationship for flexion parallel to the frontal plane. . . . .   | 42 |
| 4.12 | Prototype waist moment-displacement relationship for flexion parallel to the sagittal plane. . . . .  | 43 |
| 4.13 | Prototype waist moment-displacement relationship for rotation parallel to the transverse plane. . . . .   | 43 |
| 5.1  | Determining Static Figure Stability . . . . .   | 46 |

|     |  |    |
|-----|--|----|
| 5.2 | Example of decomposing a solid into tetrahedra . . . . .               | 48 |
| 5.3 | Calculating a Volume Integral: A Special Case . . . . .                | 61 |
| 5.4 | Inman's graph showing relation between clavicle and humerus elevations | 61 |
| 5.5 | Definitions of Joint Angles in the Shoulders . . . . .                 | 62 |

# Chapter 1

## Introduction

Human motion modeling is a useful, important part of predicting how individuals will interact with their surroundings. In the field of human factors engineering, human modeling is used to understand how an operator would accomplish some task and to design work stations and tasks that enhance the operator's effectiveness. Because it is often impossible or infeasible to create realistic prototype environments to study operator performance, a computer model of the environment and the humans who will work there is essential. Such a model would enable designers and mission developers to rehearse a mission within the computer generated environment and make refinements to both the operator's station and the mission task itself. Several iterations of planning, testing and redesign would then result in a more functional environment and a more effective mission plan.

Consider planning a task for a space mission. In this application, the micro-gravity environment and a restrictive space suit can greatly complicate the task. Given the brevity and expense of even near space missions, the value of realistic, interactive planning tools is obvious.

Human modeling is also important when studying hazardous or potentially hazardous situations. Historically, much work has been done to predict the motion of humans in sudden acceleration situations, as in a car crash or an airplane ejection

seat. Applications of computer modeling for risk assessment need not be limited to these two cases, however. One can imagine situations which are so novel that the danger to a human participant is unclear. In these situations, computer modeling could help estimate the risk.

In this thesis, I describe several approaches to human modeling using interactive computer graphics. In particular, I focus on some of the basic dynamic and kinematic properties of the human figure, and how they might be used to improve animation realism and facilitate user interaction.

Chapter 2 is a brief review of some of the work that has been done in human modeling in the fields of biomechanics and computer science. The existing anthropometric models that are the starting point of this thesis are described as well as some of the key body modeling papers that will be the bases of my joint models.

Chapter 3 describes a facility called Mover for simulating and animating the motion of a human figure under the influence of simple systems of forces and moments. The program at the heart of this facility is a modified version of DYSPAM [1, 2], a general purpose simulator for spatial mechanisms developed by R. Schaffa and B. Paul of the Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania<sup>1</sup>. This facility takes input consisting of a figure and force information, and generates an animation sequence.

For such a simulation to yield realistic motions, realistic input is required. Unfortunately, typical human motions like walking, lifting objects and other such actions are complex. And the necessary force and moment specifications required to achieve a natural looking motion may be surprisingly elusive.

In chapter 4, I focus on the other part of the simulation input - the body model. In particular, I present a simplified figure description designed especially for simulation

---

<sup>1</sup>DYSPAM is available through the Department of Mechanical Engineering and Applied Mechanics at the University of Pennsylvania, Philadelphia, Pennsylvania 19104. For further information contact Dr. Burton Paul

purposes and a model for figure joint properties based on studies in the biomechanics literature. My objective is to provide a reasonable method for modeling the properties of the human figure that could be used in conjunction with any general purpose simulator. I provide a prototype figure as an example with the intention that its “typical” joint stiffness and inertia parameters will be replaced by data collected from specific individuals when such data becomes available.

In addition to simulation, user interaction with computer models is important when planning or designing some human task. Because physically based motion algorithms are so difficult to develop, it is often convenient for the user to directly position the figure with an interactive user interface. When this technique is used, the model’s kinematic properties should be as realistic as possible. In chapter 5, I describe some algorithms for interactively positioning the shoulder complex. The shoulder, which is often casually thought of as a single ball joint, is actually a set of several bones and joints that form a fairly complicated spatial mechanism that is difficult to model as an assembly of conventional kinematic pairs (see Dvir and Berme [3] for a planar model). I model the kinematic behavior of this “joint” with a positioning algorithm that reflects the dependence of the the motion of the humerus on that of the clavicle. The algorithm is based on studies of shoulder motion and concepts from dance notation.

Extending the idea of making one joint’s position dependent on another’s, I also develop a general routine for establishing arbitrary linear relationships between sets of joint displacements. This notion of parametric relationships between graphical objects, while simple to implement, has many powerful possible applications.

Another set of tools, for calculating and interactively displaying mass and inertia properties of objects, are useful both as interactive positioning aids and as starting points for future physical simulation programs.

Finally, in chapter 6, I present conclusions about these approaches to human modeling and offer some suggestions for possible future projects.

The tools presented here are all implemented as special applications for the Jack graphics user interface in use at the University of Pennsylvania Computer Graphics Research Lab. The algorithms upon which many of them are based, however, may be applied to any system. It is my hope that the work shown here will prove to be a small step in the continuing evolution of computer graphics as applied to human modeling.



# Chapter 2

## Previous Work

Many mathematical and computer models have been developed to describe human motion in a variety of situations. One can roughly divide the work that has been done into the following categories: impact studies, gait studies, non-impact joint studies, and animation models. In each kind of study, there are elements that are very specific to the particular scenario being examined as well as more general elements that may be useful in this project. In the following sections, a brief review of the work in each category is given with some comments on the portions of the work that are relevant.

### 2.1 Crash Simulators and Ejection Seat Studies

The automotive industry conducts numerous studies each year of human response to the sudden decelerations that occur in automobile crashes. The aircraft industry conducts similar studies on human response to the sudden acceleration that occurs when ejecting from an airplane. For each of these fields, a computer simulation is typically developed that predicts the passive response of the seated figure to the large external forces, and estimates the likelihood of serious injury. Other investigators then attempt to verify the simulation results through the use of animal, cadaver, and anthropomorphic dummy experiments. Review articles by King and Chou [4] and Prasad [5] discuss some of the prominent simulation models used by industry. These

two articles are the primary sources for the discussion that follows.

King and Chou [4] describe several classes of simulators: gross-motion simulators (in two and three dimensions), and head, spine, and thorax impact models. Since my project concerns general whole-body motion, I will just review the discussion of three dimensional gross-motion simulators. The first is the three mass, 12 degree-of-freedom HSRI model developed by Robbins [6]. This model, designed primarily for the evaluation of constraint systems, calculates contact forces applied to the figure by the interior of the vehicle, which is modeled as up to 25 planes. Lap belts and shoulder harnesses may also be modeled. The three masses represent the head, torso and legs of the occupant.

A modification of the original model is reported by Robbins *et. al.* [7] in which the number of masses is increased to 6 and the number of degrees-of-freedom to 14. Collision forces between body segments are included as well as frictional forces between the body and the contact surfaces. Joint limits are also included.

The Texas Transportation Institute 3-D automobile occupant model (TTI) (Young [8]) has 12 masses and 32 degrees-of-freedom. The body segments are modeled by spheres that are connected by revolute and spherical joints. The spine is modeled with two segments and torsional springs. Joint limits are simulated by bilinear torsional viscous dampers. As with the HSRI model, the vehicle interior and restraint belts may be modeled as planar contact surfaces. However, this model contains no segment to segment collision detection.

The model by Furusho and Yokoya [9] is a 3 mass, 12 degree-of-freedom system similar to the earlier HSRI system. The body is modeled as head, torso, and legs segments with springs and dampers included to represent neck and hip stiffness. The simulation calculates seat belt loads and seat reaction and friction forces.

Another model reviewed by King and Chou is the “Superman” or UCIN model

developed by Huston et al. [10]. This model has a 12 segment, 31 degree-of-freedom body connected by revolute and spherical joints. Segments are modeled as elliptical cylinders, ellipsoids and frustums of elliptical cones. Collisions are detected between the figure and its environment, but the contact forces are not calculated.

Two three dimensional models are described in the more recent review by Prasad [5]. The first of these is the CAL3D (or CVS) model which is also mentioned by King and Chou. This model was developed by the Calspan Corporation and is described extensively by Bartz [11], Bartz and Butler [12] and Fleck et al. [13, 14]. Prasad [5] reviews a recent version of the model called CVS-IV or Version 20 which can simulate a 30 segment, 21 joint figure plus the vehicle and ground. Joints are specified as locked, pinned, ball-and-socket, or Euler joints. Up to 20 other constraints may also be specified including some segment motion specifications. Joint torques may be modeled as springs of viscous or coulomb friction mechanisms. Special elements are available for modeling tension only members (like muscles) and flexible members (like the spine). As with the other models, contact forces are calculated and restraint systems may be simulated. Prasad [5] states the program is flexible enough to be considered a general purpose articulated figure simulator. This system has been used for a number of impact studies including studies of vehicle-pedestrian impacts.

A final system is the MADYMO Crash Victim Simulation Program developed by TMO in the Netherlands. The program is described by Wismans *et al.* [15, 16]. This system allows any number of segments, but the whole figure must contain no loops, and joints may only be modeled as revolute or spherical. Joint torques are specified in tabular form. Contact reactions may be modeled as nonlinear springs, viscous damping or coulomb friction. As with the other models, restraints may be modeled, and contact forces are calculated. This system has been used in pedestrian impact studies and child restraint system studies.

## **2.2 Other Body Property Studies**

### **2.2.1 Gait Studies**

Many human body models, both whole body and lower extremity models, have been developed for the study of human locomotion. Typically, the objective of these models is to determine the joint reactions and moments that occur during walking and running. A good review of the biomechanics literature in this area is given by King [17]. King divides the research into inverse dynamics and forward dynamics studies. A forward dynamics model by Onyshko and Winter [18] models the whole body in seven segments: one for the torso and upper extremities, and three segments for each leg. The model uses a Lagrangian approach to predict the leg motion of a walking figure and a “manual” iterative approach to specifying joint torques. Initial joint angles and velocities are provided as input, joint moments during each of four phases of the gait are specified, and the resulting motion is observed. If the motion is not satisfactory, joint moments may be adjusted and the simulation run again. The ability to freely vary the moments facilitates simulating the gaits of both normal and injured subjects where particular muscle groups might be weak. Realistic results are reported, and the important physical parameters of the model are provided.

### **2.2.2 Non-Impact Joint Studies**

In order to create a computer figure that moves as a real human figure does, it is necessary to have an accurate kinematic model for the joints in the body (here “joint” is used more in the mechanisms sense than the strict medical sense). There have been a number of studies of the healthy and impaired properties of each of the numerous joints in the human body. For this project, only those studies which describe the range of motion and passive resistance properties of healthy joints are of interest. Furthermore, I focus only on certain major joints, namely the elbow,

shoulder complex, hip, and knee. Different approaches to specifying joint position and motion are also useful and are briefly reviewed. Since a comprehensive review of even the work done in the limited field described above would be beyond the scope of this project, I merely highlight a few representative papers in the recent biomechanics literature.

### **Elbow Studies**

Engin *et al.* has done a number of relevant studies of joint biomechanics. His work on the elbow [19] contains functions fitting empirical data of both voluntary range of motion of the humero-elbow complex, and the passive resistance of the elbow as a function of hyperextension angle. Both of these studies are useful in characterizing the mechanical properties of the elbow.

### **Shoulder Complex Studies**

The classic article in this area is by Inman *et al.* [20]. In this article, the ranges of motion of the different members composing the shoulder complex are described, and their interrelationships are given. Also described are the mechanics of the different force-lever mechanisms formed by the muscles and bones in the system. Action current potential measurements provide insight into what role each of the different muscles plays in each of the basic shoulder motions.

Engin [21] extends this work to include studies of the forced range of motion of the shoulder and the associated passive resistive forces and moments. Although the number of subjects in the study was very small, the shapes of the curves generated may be useful in defining elastic joint limits for simulation models.

A later study by Engin [22] reports the angular damping coefficients of the shoulder in the vertical plane as a function of upper arm position. Although the emphasis in

this paper is on the theory and method of measurement, the results given may prove useful in designing a vibration model of the shoulder.

### **Hip Studies**

Yoon and Mansour [23] provide mathematical functions fitting measurements of passive hip resistance to hip flexion and extension angles. Relations for resistance as a function of hip angle are provided for several different knee angles thus accounting for the influence of muscles that span both joints.

### **Knee Studies**

A thorough description of the envelope of motion for the knee joint is given by Blakevoort *et al.* [24]. This is an *in vitro* study of the range of motion in flexion and in tibial rotation. External forces are applied to the specimen knee and the resulting force-displacement plots are provided. Since the study is performed on cadaver knees, the actual values of the torques reported at the joint limits might be in question, but the general shape of the force-displacement curve may be similar to that of a live subject. It will probably be necessary to compare these results with other studies to verify the validity of the method.

### **General Joint Models**

Most joints in the human body have some amount of compliance even in directions other than those of their primary action or motion. Although this compliance is slight, a truly thorough model would consider practically every joint as a six degree of freedom joint with some allowable translation and some allowable rotation in every direction. Typically, though, the range of motion in translation is slight, and the amount of compliance in directions that are not usual for the joint is also fairly

small. Hence most simulations that model joints and joint limits make simplifying assumptions about the “allowable” motion of the joints. A thorough discussion of the different models that are used is contained in a paper by Kinzel and Gutowski [25]. The most common simplifications are hinge or revolute joints for the knees and elbows, and ball and socket or spherical joints for the shoulders and hips.

Another useful topic in the literature is the specification of coordinate systems appropriate for describing joints. Grood and Suntay [26] have written a number of articles suggesting a scheme similar to Euler angles, but having the advantage that relative translations could also be represented. Perhaps even more importantly, in the proposed scheme (unlike in the usual euler angle system), the order of the relative rotations of the moving segments does not need to be specified. The particular paper cited applied the “joint coordinate” system to the knee, but other papers by the same authors consider the spine and other joints in the body.

## **2.3 The Graphics Environment**

### **2.3.1 Animation Models**

Human figures developed for computer animation applications are as many and varied as those developed for impact simulations. The figures reviewed by Dooley [27], for instance, have a large amount of articulation and anthropometric accuracy and flexibility. However, they are lacking in the physical properties that would be needed for dynamic simulations. Conversely, systems which provided animation post-processors for existing human motion simulators (like those described earlier) are often lacking in anatomical complexity or the ability to model a variety of different dynamic situations (see Wilmert [28]).

More recently, work has been done to incorporate dynamic directly into animation systems. And appropriate human body models have developed to accompany

these systems. Wilhelms [29] used an 18 degree-of-freedom figure to demonstrate her dynamically driven animation system, but the figure was a very simple one from an anthropometric and anatomical viewpoint. It did contain joint limits that were modeled spring and damper systems, but apparently no effort was given to establishing a force-displacement behavior that mimicked the actual human body.

Work by Girard [30] on animal and human locomotion also places a heavy emphasis on motion controlling algorithms and less emphasis on the accuracy of the models the algorithms drive. Physical parameters such as mass distribution within the figure are used in the algorithms, but their values apparently are not related to those of real animals.

For accuracy and care in developing anthropomorphic human figures, one should consider the models developed in the Computer Graphics Research Lab at the University of Pennsylvania. A technical report by Grosso *et al.* [31] describes the latest developments in generating anthropometrically accurate figures of different sizes based on population information. These figures have 31 segments and 42 degrees-of-freedom. Kinematic joint limits are available and based on anthropometry data. Until recently, however, the ability to use these models in animations that were physically based was absent.

### **2.3.2 Graphics Lab Software**

All tools developed in this project are intended to be extensions of existing software developed at the University of Pennsylvania Computer Graphics Research Laboratory. Two important parts of that body of software are Peabody and Jack. Peabody is a graph-structured representation for articulated figures (see [32] for details). It is a language for representing figure information including all figure location, joint, and segment connectivity information, as well as other physical attributes of the figure being described. The data files (called “figure files” in this thesis) are structured,



much like a programming language. This facilitates editing and understanding the data. These files are read by a parser that converts the syntax and data into the computer's internal data representation.

Jack is a graphic user interface program and a library of subroutines that serve as the foundation upon which various applications may be built (see Phillips [33]). Jack provides an interactive 3D graphics environment for modeling, displaying and manipulating articulated figures. A mouse and nested menus provide easy user interaction with existing utilities. Jack runs on a Silicon Graphics IRIS graphics workstation and provides capabilities such as animation, real-time rendering and real-time 3D manipulation of represented objects.

## Chapter 3

# Mover: A Dynamics Driver for the Jack Environment

Mover is the name given to a collection of computer programs which allow the general purpose mechanism simulator DYSPAM to act as a preprocessor for the Jack graphics environment. A situation may be set up using the Jack user interface, saved as a Peabody representation, and shipped to Mover which will in turn run a simulation and generate an animation file which can then be played back in the Jack environment. This sequence of actions greatly simplifies the design of a simulation set-up and allows easy blending of keyframe animation with simulation results. Also, attaching DYSPAM in this way to a powerful graphics system grants the benefits of three-dimensional visualization of both the initial conditions and the resulting motion. The following sections will describe the problems involved in creating this system, and the resulting conversion programs themselves.

### 3.1 Data Representations

Both Jack and DYSPAM represent articulated figures, and both have similar notions of degrees of freedom and local and global coordinate systems, but the formats of their data representations are radically different. This difference was the main obstacle in achieving an integration of the two programs without extensively changing either.

### 3.1.1 DYSPAM files

DYSPAM, as it is used our lab, takes 3 input files<sup>1</sup>: structure, ndyspamfile, and jackforce. A brief overview of the contents of each file is given with a sample file for illustration. For more detailed descriptions of the input format see Schaffa [1] or Strassberg [34].

#### Structure

Structure (see figure 3.1) contains the basic information for controlling the simulation. The first line contains the number of bodies, number of joints, and the number degrees of freedom in the system. The next line contains the number global forces, local forces, global moments and local moments applied to the body. The remainder of the file contains flags and numerical integration and equation solving parameters. Also included, is the number of rotational springs in the system and time information for integration step size and the time interval between iterations to be written as output.

```
1, 1, 3    ——— number of bodies, joints, degrees of freedom
1 0 0 0    ——— local forces, global forces, local moments, global moments
0, 0, 3    ——— flags and number of rotational springs
0, 0      }
0         } ——— more flags
0.000000, 0.10000, 30.0, 0.300000 start time, time step, stop time, animation step
0.00000001, 0.00001, 0.000001, 7 }
10.0 ,1.0 ,0.1/                    } numerical procedure parameters
```

Figure 3.1: DYSPAM Input File: structure

#### Ndyspamfile

Ndyspamfile (see figure 3.2) contains the mechanism description. Information is arranged in 6 sections (or tables):

---

<sup>1</sup>A special file called “optional” may be also be used. This file specifies non-linear springs and is described in Chapter 4.

- body joint table - a list of connectivity relationships and joint types connecting pairs of segments.
- joint triad table - the location and orientation of joint coordinate frames in terms of the segment's local reference frame
- mass and inertia information - a list of each segment's mass, center of mass site, and principal moments of inertia.
- point of interest table - a list of site locations and the segments to which they belong.
- spring parameters - linear spring and damping constants, rest angles, range of hysteresis, and the degree of freedom with which the spring is associated.
- initial conditions - initial displacements and velocities for all the figure's degrees of freedom.

```

1 0 1 7 ----- body joint table
0 1
  0.00   0.00  -1.00   0.00
  0.00   1.00   0.00  120.00
  1.00   0.00   0.00   0.00
1 1
  1.00   0.00   0.00   0.00
  0.00   1.00   0.00   0.00
  0.00   0.00   1.00   0.00
----- joint triad table
1 3 25.000000 250.000000 50.000000 50.000000 ----- mass and inertia data
1 1 0.00,0.00,0.00
2 1 150.00,0.00,-15.00 } ----- point of interest table
3 1 75.00,0.00,0.00
1 1000.000000 600.000000 0.000000 1
1 500.000000 600.000000 0.000000 2
1 800.000000 600.000000 0.000000 3
----- spring parameters
1 0.349066 0.698132 0.000000 0.00 0.00 0.00 ----- initial conditions

```

Figure 3.2: DYSPAM Input File: ndyspamfile

### Jackforce

The last of the input files, jackforce (see figure 3.3), contains a list of the local forces, global forces, local moments and global moments applied to the system. Each item is listed with the number of a point of interest where it is acting (from the point of

interest table in ndyspamfile) and its global x, y and z components. These forces are constant only, with time varying forces requiring a user-specified routine called FORCES within the DYSPAM source code.

```
1 3 100.00 0.0 0.0
```

Figure 3.3: DYSPAM Input File: jackforce

### 3.1.2 Peabody figure files

Much of the same information contained in the DYSPAM input files is also contained in the Peabody figure file. However, the same information takes a dramatically different form as can be seen in figure 3.4. Here, the format uses a data representation language which can be interpreted by the Peabody parser (see Phillips [32]). The information is then stored internally in an extensive data structure.

```

figure {

    segment floor {
        psurf = "floor.pss";
        site prox ->location = xyz(0,0,0) * trans(0,0,0);
        site dist ->location = xyz(-45deg, -90.00deg , -45.00deg) *
            trans(0,120cm,0);

        mass = -1.0;
    }

    segment link1 {
        psurf = "link1.pss";
        site prox {
            location = xyz(0,0,0) * trans(0,0,0);
        }
        site dist ->location = xyz(0,0,0)
            * trans(150.00cm,0.00cm,-15.00cm);

        site CM{
            location = xyz(0,0,0) * trans(75.00cm,0.00cm,0.00cm);
            globalforce = (100.0, 0.0, 0.0);
        }
        mass = 25.0;
        inertia = (50,50,50);
    }

    root = floor.prox;

    joint shoulder {
        connect floor.dist to link1.prox;
        type = R(1,0,0)*R(0,1,0)*R(0,0,1);
        displacement = (0deg, 40deg, 20deg);
        stiff = ( 100, 1000, 1000);
        rest = (0, 0 , 0);
    }
}

```

Figure 3.4: Peabody Figure File

Essentially, a figure description is a structure composed of several sub-structures. Each structure and substructure has an identifier such as “segment” and a name, such as “floor”.

The segment sub-structure contains all the information concerning a particular segment (a.k.a. body) in the system. Psurf refers to a named file containing the geometry (i.e. three dimensional shape) information for display purposes. The location of the origin in the definition of the psurf defines the local coordinate system of the segment. Sites are equivalent to “Points of Interest” in the DYSPAM nomenclature. Each site is itself a “sub-sub-structure” with position and orientation fields. Position is specified with the identifier “trans” and orientation is specified with the identifier “xyz”. “xyz(10deg,20deg,30deg)” indicates that the site in question is in an orientation that can be achieved by rotating the local coordinate frame first 10 degrees about the x-axis then 20 degrees about the rotated y-axis and finally 30 degrees about the doubly rotated z-axis. Sites may be named in any fashion convenient to the user, but the name “CM” is reserved for that site locating the center of mass of the segment. The segment’s mass and inertia information are indicated in their appropriate fields. As in DYSPAM, the Peabody representation assumes the local coordinate system of the segment is parallel with the principle inertial axes of the body. Finally, each site has global and local force and moment fields in which constant external forces and moments may be specified.

The joint sub-structure contains the connectivity information and all other joint related parameters. Each joint connects two and only two segments as indicated by the “connect” specification. “connect floor.dist to link1.prox” means that this joint connects the site named “dist” that is a part of the segment named “floor” to the site named “prox” that is a part of the segment named “link1”. The type specification indicates the number and type of the degrees of freedom of the joint. “R(1,0,0)” indicates a rotational degree of freedom about the local x-axis. “T(1,0,0)” indicates a translational degree of freedom along the x-axis. By chaining a sequence of these degrees of freedom together, a number of different kinds of mechanical joints may be specified. Each joint also has fields for specifying joint limits and spring information such as a spring constant, damping constant and rest angle.

The identifier “root” indicates which site can be considered attached to the ground

(i.e. unmoving) segment.

From this discussion it should be apparent that the DYSPAM input files and the PEABODY figure files contain much the same information. The primary differences between them can be summarized:

- DYSPAM files contain time related information such as velocity and simulation start and stop times.
- DYSPAM files contain numerical procedure parameters.
- DYSPAM files contain flags for invoking more complex procedures.
- Peabody files contain references to figure geometry.

Fortunately, most of the differences may be safely ignored. DYSPAM, having no display capabilities of its own has no need for the psurf geometry files. And flags and numerical procedure parameters may be set in advance and assumed to be the same for all cases that are expected to be encountered. The only remaining considerations are the data related to time and velocity. These are only a few values, so it is not too great a burden to specify them at the time the simulation is called.

What remains is the task of converting those elements that *are* in common from one format to the other.

### 3.1.3 Joint Transformation Conventions

To complicate the conversion process, there exist within DYSPAM some restrictions on the definition of joint triads associated with certain kinematic pairs. These restrictions are absent from the Peabody representation. First, DYSPAM expects all revolute joints to be defined so that the axis of rotation is the positive z-axis. Peabody, however, allow revolutes to be defined about any axis, be it a positive or negative coordinate axis, or even some other arbitrary axis.

Second, DYSPAM defines its spherical joints using a Z-Y-X Euler angle convention, so any initial position or output position of a spherical joint is defined in terms of this convention. Peabody, however, allows any sequence of rotations as its definition of a spherical joint. So proper interpretation of the figure file by DYSPAM and proper creation of an animation file, both needed the ability to convert from or to an arbitrary relative rotation sequence.



Other joint types have restrictions also, but since these are the only ones that occur in the human body model, these are the only ones that are accommodated.

## **3.2 Ernest: Converting the Input**

Ernest was first written by Lee and Chu <sup>2</sup> as a basic interface between Jack and DYSPAM. The approach was simple: read a Peabody figure file, parse it with the Peabody parser and then query the internal data representation for each item required for the DYSPAM input files, and write them to the appropriate file. Unfortunately, the program was not very flexible as it could not handle joints other than the revolute type, and those joints were required to be specified in the DYSPAM fashion, with the rotation being about the local z-axis. However, the basic work of interpreting the Peabody data and counting the number of joints, sites, and segments and arranging the data so that it could be output as DYSPAM input files was accomplished.

### **3.2.1 Converting Revolute Joints**

In consultation with Phillips<sup>3</sup> we developed a scheme for handling the restriction on the definition of revolute joints. The objective was to allow the existing freedom in the Peabody description while still passing only “legal” joints to DYSPAM. The solution was to internally redefine all revolute joints that were not already defined about the z-axis so that they were, and then writing that description out to the DYSPAM input files. The function used was called “zifyjoint”. The algorithm is outlined in figure 3.5.

### **3.2.2 Converting Spherical Joints**

The conversion of spherical joints presented a slightly more complicated problem. Since Peabody allows spherical joints to be composed of three sequential rotations about arbitrary axes, some arbitrary set of initial displacements had to be converted to the DYSPAM convention. As an additional complication, it was realized that the displacements produced by DYSPAM as output from the simulation would have to be

---

<sup>2</sup>This program was provided to me by one of the authors, Phillip Lee

<sup>3</sup>Cary B. Phillips, University of Pennsylvania Department of Computer Science, 1989

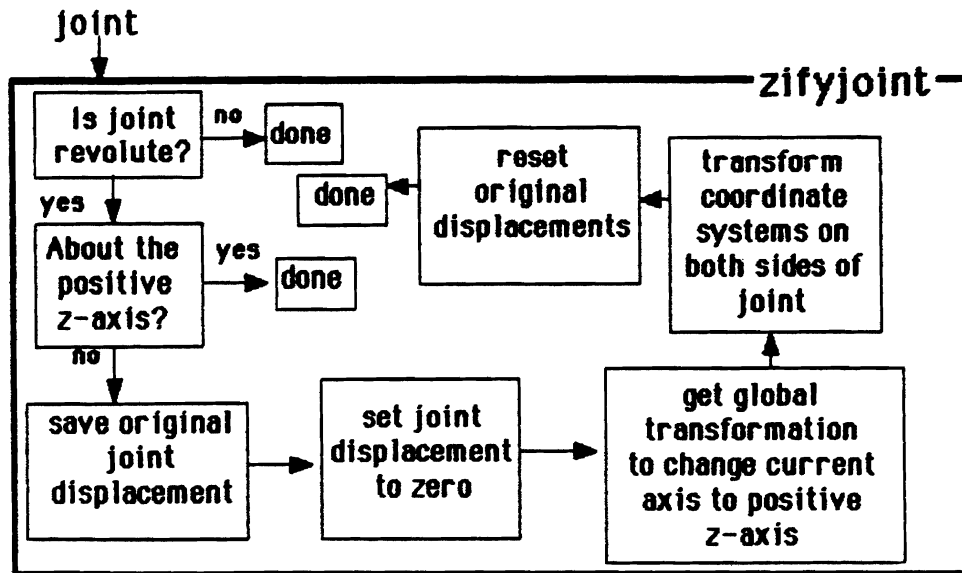


Figure 3.5: Zifyjoint: Algorithm for Handling Restriction on Revolute Joints

converted back to the original Peabody rotation convention so the animation system would be able to display the sequence of positions accurately. So whatever solution was used would have to be reversible. Once again, the objective was to pass equivalent information to DYSPAM while imposing few, if any restrictions on the Peabody joint definitions.

The strategy for achieving the conversion was simple. First, the data structure would be read to determine what sequence of rotations was being specified by the Peabody figure file. With that information, the three initial joint displacements can be interpreted correctly, and an appropriate rotation matrix calculated. Then the elements of the rotation matrix can be used to find the equivalent joint displacements in the Z-Y-X system (the DYSPAM rotation convention).

If  $\phi$ ,  $\theta$  and  $\psi$  are relative rotations about the z, y and x-axes, respectively, then

the corresponding rotation matrix is given by<sup>4</sup>:

$$\mathbf{R} = \begin{bmatrix} \cos \theta \cos \phi & \sin \psi \sin \theta \cos \phi - \cos \psi \sin \phi & \cos \psi \sin \theta \cos \phi + \sin \psi \sin \phi \\ \cos \theta \sin \phi & \sin \phi \sin \psi \sin \theta + \cos \psi \cos \phi & \cos \psi \sin \theta \sin \phi - \sin \psi \cos \phi \\ -\sin \theta & \sin \psi \cos \theta & \cos \psi \cos \theta \end{bmatrix}$$

Z-Y-X Euler angles can be found for any rotation matrix by observing the relationships of certain elements of the above matrix.

$$R_{2,1} = \cos \theta \sin \phi \quad (3.1)$$

$$R_{1,1} = \cos \theta \cos \phi \quad (3.2)$$

$$R_{3,2} = \sin \psi \cos \theta \quad (3.3)$$

$$R_{3,3} = \cos \psi \cos \theta \quad (3.4)$$

$$R_{3,1} = -\sin \theta \quad (3.5)$$

From these relationships we can find  $\phi$ ,  $\psi$  and  $\theta$  except in two special cases. In general,

$$\phi = \text{atan2}(R_{2,1}, R_{1,1}) \quad (3.6)$$

$$\psi = \text{atan2}(R_{3,2}, R_{3,3}) \quad (3.7)$$

$$\cos \theta = \sqrt{R_{2,1}^2 + R_{1,1}^2} \quad (3.8)$$

$$\theta = \text{atan2}(-R_{3,1}, \cos \theta) \quad (3.9)$$

The two special cases occur when  $\cos \theta = 0$ . This can happen in two ways. If  $-R_{3,1} = \sin \theta = 1$  then we know  $\theta = \pi/2$ . In this case, we solve for the difference between  $\phi$  and  $\psi$ :

$$R_{2,3} = \cos \psi \sin \phi - \sin \psi \cos \phi = \sin(\phi - \psi) \quad (3.10)$$

$$R_{1,3} = \cos \psi \cos \phi + \sin \psi \sin \phi = \cos(\phi - \psi) \quad (3.11)$$

$$(\phi - \psi) = \text{atan2}(R_{2,3}, R_{1,3}) \quad (3.12)$$

$\theta = \pi/2$  corresponds, in this case, to a 90 degree rotation about the y-axis which places the x-axis in direct opposition to the original z-axis thus effectively reducing

---

<sup>4</sup>Throughout this work I will use the robotics rotation matrix convention in which the columns of the matrix may be interpreted as vectors along the rotated coordinate axes referred to the unrotated axes. The computer graphics convention is to use rows instead.

the number of degrees of freedom. The single degree of freedom replacing  $\psi$  and  $\phi$  in this case is the difference  $\phi - \psi$ .  $\phi$  or  $\psi$  may therefore be set arbitrarily. We establish the relationship:

$$\phi = -\psi \tag{3.13}$$

So the resulting displacements are:

$$\phi = -\psi = \frac{1}{2} \text{atan2}(R_{2,3}, R_{1,3}) \tag{3.14}$$

$R_{3,1} = 1$  indicates the other special case,  $\theta = -\pi/2$ . By reasoning similar to the previous case, we find

$$\phi = \psi = \frac{1}{2} \text{atan2}(R_{2,3}, R_{1,3}) \tag{3.15}$$

This algorithm is implemented in a function called “tozyx” which is called by the section of Ernest which handles initial conditions.

### 3.3 Pebble: Converting the Output

The output of DYSPAM essentially consists of a sequence of time values each followed by a list of joint displacements specifying the configuration of the system at that time. The joint displacements correspond, of course, to angles as defined in DYSPAM conventions. The output conversion program’s task is to replace the joint displacements in the output with displacements as defined in the particular figure file that was the original source for the system description.

Accommodating the differences in convention for revolute joints was simply a matter of checking if the Peabody definition of a positive displacement was in a counter-clockwise direction (i.e. defined about a positive coordinate axis) or a clockwise direction (i.e. defined about a negative axis). Since DYSPAM revolute joints are always positive in a counter-clockwise direction, those defined in the opposite way in the Peabody definition required a sign change for their displacement.

Converting spherical displacements presented a problem that was solved by imposing a restriction on the kinds of spherical joint definitions that could be converted from the DYSPAM convention. As in Ernest, we needed a mechanism that would take a rotation matrix, and produce a set of angles representing that rotation in some Euler angle-like convention. However, without restricting the sequence to be about

mutually perpendicular axes, it would be very difficult to apply an algorithm similar to the one in the previous section to handle the conversion. We decided that a slight restriction would not significantly reduce the user's convenience when designing Peabody figures. Sequences of rotations would have to be about coordinate axes or negative coordinate axes. Arbitrary axes within the local joint coordinate system would not be allowed. However, the location and orientation of the joint triads themselves could always be defined arbitrarily so the restriction on the kinds of joints that could be used was purely syntactical.

Not counting definitions of rotations about negative axes, there are twelve sequences of rotations that meet the above restriction and are capable of producing three rotational degrees of freedom. They are:

1.  $X \rightarrow Y \rightarrow Z$
2.  $Z \rightarrow X \rightarrow Y$
3.  $Y \rightarrow Z \rightarrow X$
4.  $X \rightarrow Z \rightarrow X$
5.  $Z \rightarrow X \rightarrow Z$
6.  $Y \rightarrow X \rightarrow Z$
7.  $Z \rightarrow Y \rightarrow X$
8.  $X \rightarrow Y \rightarrow X$
9.  $Z \rightarrow Y \rightarrow Z$
10.  $Y \rightarrow X \rightarrow Y$
11.  $Y \rightarrow Z \rightarrow Y$
12.  $X \rightarrow Z \rightarrow Y$

For each of these cases, it is simple to find the angles in the given convention that correspond to a rotation matrix. This is the heart of Pebble. The rest of the algorithm is outlined in figure 3.6.

### 3.4 Integrating the Programs

Mover is a shell program that makes the sequence of the conversion programs and the simulator itself a little more manageable. Ernest takes as input a Peabody figure file and time information provided by the user and produces DYSPAM input files structure, ndyspamfile and jackforce. DYSPAM can then use those files, run a simulation

1. Read the original figure file and parse it to get the data structures.
2. Step through the DYSPAM output file time step by time step.
3. For each time step:
  - a. Normalize the time (necessary for animation)
  - b. Step through each joint in order as defined in the figure file.
    - i. identify the joint type (revolute, prismatic, etc.)
    - ii. check to make sure joints are defined according to the restriction
    - iii. convert the joint angles from the DYSPAM file base on the type of joint (assume all rotations are about positive axes)
    - iv. change the signs of those displacements about negative axes.
    - v. Write to the output file (known as an action file)
  - c. next joint
4. next time step

Figure 3.6: Pebble: Converting DYSPAM output to Jack input

and produce a set of output files including the “animation” output file called out.e. Pebble then takes out.e and the original figure file and generates a Jack compatible animation file (known as an action file). Since the whole process requires only a figure file as input, it makes sense to hide the internal workings of the sequence by creating a macro that handles all the other files for the user (see figure 3.7). This is the function of Mover.

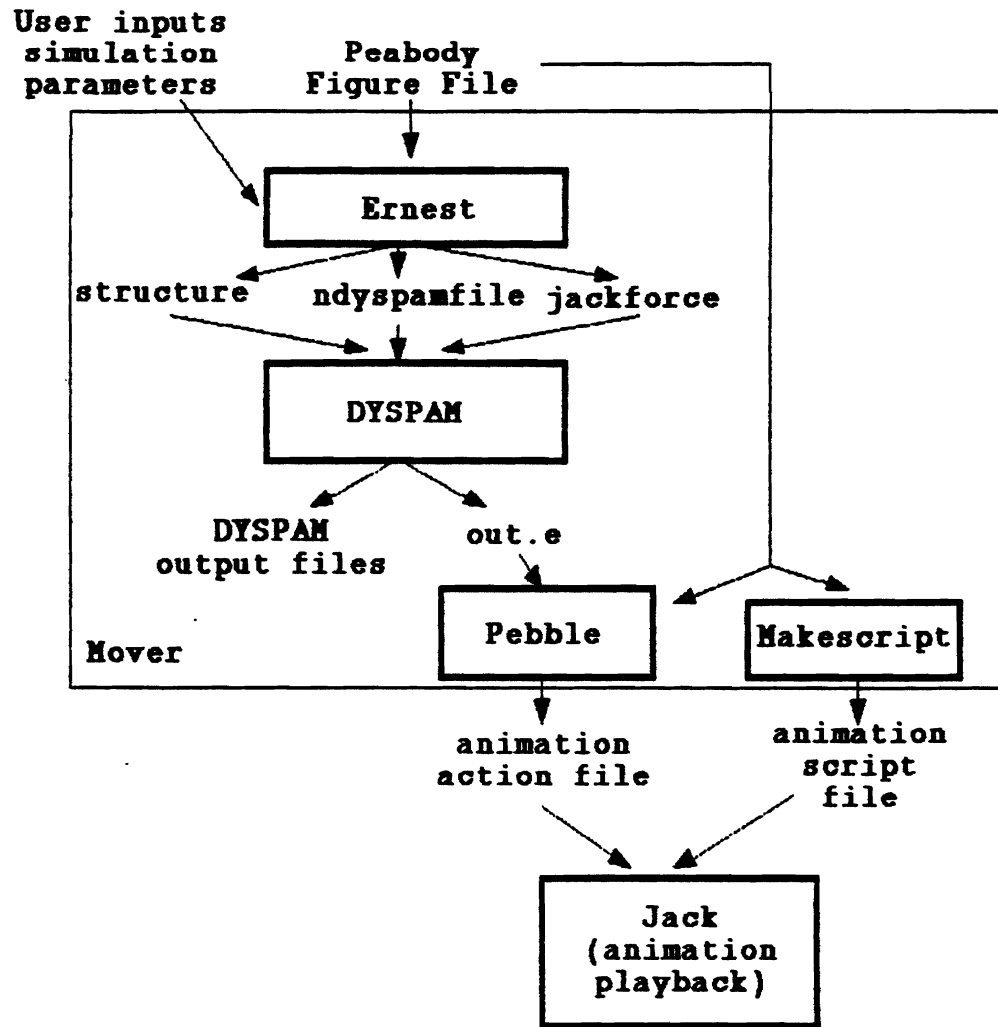


Figure 3.7: Information Flow of Mover

# Chapter 4

## The Body Model

In this chapter, I continue the development of the anthropomorphic figure described in chapter 2. That figure has the shape and size and joints of a human, but few of the of the properties necessary for a dynamic simulation. It is like a mannequin that can be posed, but does not move as a human would under the influence of external forces. The objective of this portion of the project is to add sufficient information to the model so that the new model would be to the old as a cadaver is to a mannequin. That is, a model of a body possessing accurate mass and inertia information, and having joints that behave realistically - with motion limits and elastic and damping properties. This model then would be ready for the next generation of refinements that could include motion algorithms and strength information to generate naturalistic human actions.

### 4.1 Modeling Joint Properties

One of the most obvious differences between the motion of a human and a human-like model is the limits on the human's range of motion. That is, elbows of humans bend easily only through a limited range of angles, whereas a simply hinged figure's joints have no such limits. Joint limits in JACK are specified as kinematic restrictions on the positioning of joints both internally, and via user interaction. This feature



prevents the user from placing the figure in a configuration that would be impossible (or extremely uncomfortable) for a real human. The task here is to extend this idea to incorporate these joint limits into the Mover simulation system. This amounts to characterizing the force versus displacement relationships of typical body joints and applying these relationships within the simulation driver (in our case, DYSPAM). In this section, I will describe my scheme to represent passive resistive moments at joints, and how they are applied to the model.

#### **4.1.1 Active and Passive Joint Properties**

Joint limits can be thought of in two ways. One can consider a limit to be the displacement at which a human can no longer continue to move his own joint in a particular direction. This kind of limit I will call an “active limit”. It is reasonable to assume that this kind of limit is easily measured from live subjects and probably is a function of both the individual’s strength and her suppleness.

A joint limit can also be thought of as the displacement at which the resistive forces of stretched body tissues associated with the joint achieve some arbitrary value. Often this value corresponds to the displacement at which a live subject would begin to feel discomfort. This kind of limit I will call a “passive limit” since the limit is assumed to be determined by measuring a subject that is not consciously attempting to resist the motion of his joint. Passive limits can be measured on both live subjects and cadavers (although it is unclear whether results obtained from cadavers are valid for live subjects). This limit is independent of the subject’s strength although strength may be a predictor of the limit’s value.

Put simply, an active limit is the angle to which a human can bend his own joint, and a passive limit is the angle to which some external force can force the human’s joint.

In the context of setting purely kinematic limits on motion, either active or passive

limits may be employed. The choice would depend on whether the motion is intended to be generated by the figure itself or by some external impetus. However, in the context of dynamic simulation, it is not the value of the limits that are important but the relationship between the joint displacement and the resistive moment associated with it. The resistance may be either passive or active. Only passive resistances will be considered here since active resistance implies use of strength and strength modeling is beyond the scope of this project<sup>1</sup>.

#### 4.1.2 Selecting a Stiffness Function

To adequately model the relationship between resistive moments and joint displacements, it is necessary to have a method for storing different relationships in a file and then using that stored information to generate appropriate moment values for given displacements. There are two ways of accomplishing this: by table or by function. If a table is used then the whole table of data must be stored and retrieved, and an interpolation routine would have to be used to determine values between entries in the table. The values generated could be extremely faithful to the actual data if enough points are included in the table.

The other strategy is to fit some kind of function to the relationships in advance and then just store and retrieve the parameters of the function. In this case, all the values would be generated, and how faithful that generated data would be to the original data would depend on how good the fit was. An advantage of this strategy is the convenience of having uniform representations with small sets of parameters.

Both strategies have merits, but I use fitted functions rather than tables because the implementation is slightly simpler, and the fits seem very good. Each set of joint data gleaned from the biomechanics literature is fitted to a cubic polynomial. The

---

<sup>1</sup>It is possible that the addition of a strength model to the passive resistance model described here might be capable of replicating in simulation active joint limits measured from actual human subjects.

cubic polynomial is a good choice because it is widely used in crash simulators and also because its shape can closely match many of the moment-displacement curves in the literature. Comparisons of the fitted functions and the actual data will appear in later sections. Data is fit to the polynomial using a least-squares method.

### 4.1.3 Stiffness Properties of Spherical Joints

Representing the stiffness behavior of spherical joints is particularly difficult since most empirical studies only examine moment-displacement relationships about each of the usual coordinate axes with little or no data collected for rotations about oblique axes. For this reason, the model developed for this class of joint can at best be considered only a reasonable approximation of actual joint behavior.

Spherical joints are modeled as a directionally weighted average of the influences of three non-linear spring-dashpot systems, one for each degree of freedom. Each equation relates a displacement  $\alpha$  to a moment  $M$  :

$$M_x = c_0 + c_1\alpha_x + c_2\dot{\alpha}_x^2 + c_3\alpha_x^3 + b_v\dot{\alpha}_x + b_{\text{coul}}\text{sgn}(\alpha_x) \quad (4.1)$$

$$M_y = c_0 + c_1\alpha_y + c_2\dot{\alpha}_y^2 + c_3\alpha_y^3 + b_v\dot{\alpha}_y + b_{\text{coul}}\text{sgn}(\alpha_y) \quad (4.2)$$

$$M_z = c_0 + c_1\alpha_z + c_2\dot{\alpha}_z^2 + c_3\alpha_z^3 + b_v\dot{\alpha}_z + b_{\text{coul}}\text{sgn}(\alpha_z) \quad (4.3)$$

Here, the coefficients  $c$  define the moment-displacement relationship of the joint about the different coordinate axes,  $b_v$  is a viscous damping coefficient and  $b_{\text{coul}}$  is a coulomb friction element. Typically, these relations are derivable from available empirical results with different coefficients applying to motion about different axes.

Any change in orientation of one segment relative to another can be expressed as a single angular displacement  $\alpha$  about some axis  $\mathbf{A}$ <sup>2</sup>. If the vector  $\mathbf{A}$  is defined as a unit vector, then it is possible to devise a weighting scheme that is based on the

---

<sup>2</sup>Goldstein [35] has a good description of the method for finding an axis-angle representation from a set of euler angles.

relationship between the direction of the rotation axis  $\mathbf{A}$  and the joint coordinate system. The scheme used in this model is:

$$\mathbf{M} = A_x M_x(\alpha)\mathbf{i} + A_y M_y(\alpha)\mathbf{j} + A_z M_z(\alpha)\mathbf{k} \quad (4.4)$$

Here,  $M_x, M_y, M_z$  are the three spring systems described in equations 4.1, 4.2, 4.3,  $A_x, A_y,$  and  $A_z$  are the components of the rotation axis and  $\mathbf{i}, \mathbf{j}$  and  $\mathbf{k}$  are the usual Cartesian unit vectors.

Notice that when the rotation axis  $\mathbf{A}$  lies along a coordinate axis, the equation above reduces to the single component moment equation for that axis. There is no guarantee that values for rotations about non-coordinate axes will be accurate, but the estimate seems reasonable.

#### 4.1.4 Stiffness Properties of Revolute Joints

Revolute joints are typically characterized by a region in the range of motion that is virtually free of influence from joint stiffness. I will call this region the deadzone. In order to adequately model this behavior I divide the range of motion into three regions in which the moment-displacement behavior is defined by:

$$M = \begin{cases} c_0 + c_1\alpha + c_2\alpha^2 + c_3\alpha^3 + b_v\dot{\alpha} + b_{\text{coul}}\text{sgn}(\alpha) - M_{\alpha_l} & \text{for } \alpha < \alpha_l \\ 0.0 + b_v\dot{\alpha} + b_{\text{coul}}\text{sgn}(\alpha) & \text{for } \alpha_l \leq \alpha \leq \alpha_u \\ c_0 + c_1\alpha + c_2\alpha^2 + c_3\alpha^3 + b_v\dot{\alpha} + b_{\text{coul}}\text{sgn}(\alpha) - M_{\alpha_u} & \text{for } \alpha > \alpha_u \end{cases} \quad (4.5)$$

$$M_{\alpha_l} = c_0 + c_1\alpha_l + c_2\alpha_l^2 + c_3\alpha_l^3 \quad (4.6)$$

$$M_{\alpha_u} = c_0 + c_1\alpha_u + c_2\alpha_u^2 + c_3\alpha_u^3 \quad (4.7)$$

Where  $\alpha_l$  and  $\alpha_u$  are the boundaries of the deadzone.

For revolute joints  $\alpha$  is defined as the difference between some rest value  $\alpha_0$  and the actual displacement.

```

1 1 _____ joint number, axis number
0.0 2.4365e09 0.0 8.4168e09 _____  $c_0, c_1, c_2, c_3$ 
0.0 0.0 _____  $b_v, b_{coul}$ 
0.0 0.0 0.0 _____  $\alpha_0, \alpha_l, \alpha_u$ 
1 2 _____ joint number, axis number
7.9322e08 -4.658e09 8.617e09 -5.114e09
0.0 0.0
0.0 0.0 0.0
1 3
0.0 4.3931e07 0.0 -3.098e08
800000 0.0
0.0 0.0 0.0
9 3
-155000000 845000000 -1240000000 492000000
800000 0.0
0.0 0.0 1.553

```

Figure 4.1: Data format for specifying non-linear springs

#### 4.1.5 Non-linear Springs: Data Representation

The specification for these formulations for non-linear springs are passed to DYSPAM via a special input file called optional. The file is so named because either linear or non-linear springs can currently be specified, with the linear springs being standard and the non-linear springs optional. Within the file, is a listing for the parameters of each degree of freedom employing a rotational spring. Springs for spherical joints and revolute joints are specified in the same format even though some of the parameters are not used for both. The specifications for  $\alpha_u$ ,  $\alpha_l$  and  $\alpha_0$  are simply ignored by the routine that calculates the spring forces for spherical joints. An example of the format is given in figure 4.1. The units are in the cgs (cm-gram-second) system.

## 4.2 The Prototype Simulation Figure

It is impossible to characterize all humans with a single model. Humans, even “typical” ones, vary widely in stature, mass distribution, and suppleness. In this section,

I propose a prototype simulation figure as an example of what a figure specifically designed for simulation studies might be like. The values used for the figure's body parameters, such as segment lengths, mass properties and so on are based on typical anthropometric values, though not necessarily from the same population of subjects. The model, therefore, should be considered only a template for more accurate, or carefully collected body parameters.

The prototype figure is a simplified version of the 50th percentile male body definition used at the University of Pennsylvania Computer Graphics Research Lab. It has 10 segments and 10 joints with 26 degrees of freedom. The simplification is necessary to prevent excessive computation time during simulation. The Peabody definition is given in the appendix. The geometry for the figure consolidates psurfs from some of the smaller segments into the larger segments thus achieving a figure that is structurally much simpler than the original, but whose appearance is about the same. For example, the foot and toe psurfs are consolidated into the lower leg. The resulting figure has the following segments:

- torso segment - includes head and neck and clavicles
- upper arm segments - (left and right)
- lower arm segments - (left and right) includes hands
- lower torso segment
- upper leg segments - (left and right)
- lower leg segments - (left and right) includes feet

Along with the reduction in numbers of segment is a reduction in joints. Only the major joints (elbows, knees, hips, shoulders and waist) remain in this simplified figure. The following sections discuss the prototype moment-displacement relationships and the studies upon which they are based.

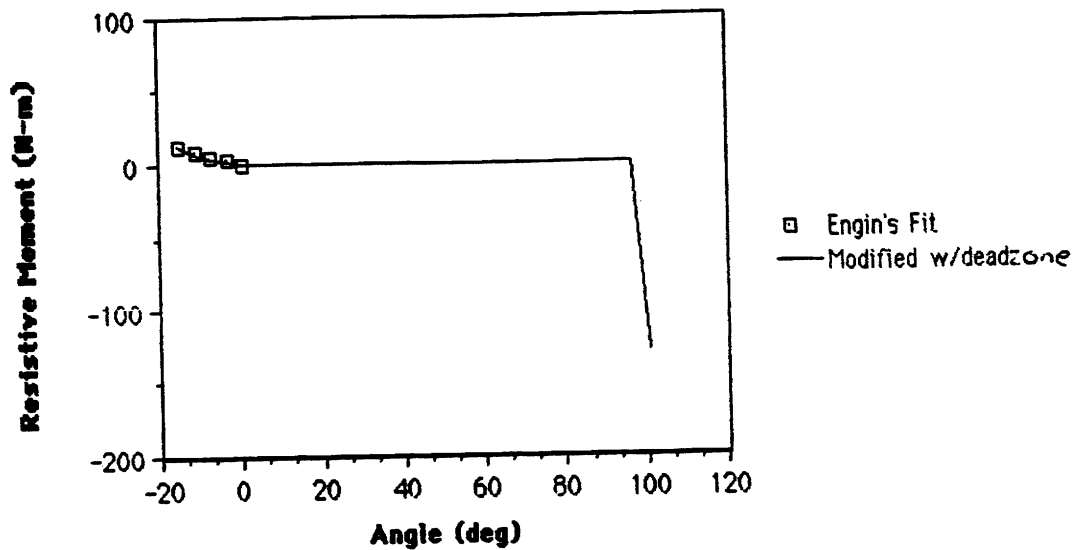


Figure 4.2: Elbow moment-displacement relationship: Engin's elbow data and prototype elbow model

#### 4.2.1 Elbow Springs

The prototype elbow is based on a study by Engin and Chen [19]. Engin measured the moment-displacement relationship of ten healthy males in elbow extension and hyperextension. The results of the measurements were then fit to cubic polynomials. The mean values for the coefficients are used as the prototype values. Figure 4.2 shows the resulting polynomial and the modeled behavior that includes a resistance-free region. The values for  $\alpha_u$  and  $\alpha_l$  are estimates.

#### 4.2.2 Knee Springs

The prototype knee is based on the technical report accompanying the CAL3D crash simulator [13]. The moment-displacement relationship given in that report was obtained by measuring torque values from a Sierra 292-1050 crash dummy. It is unclear how well such results would correlate with results obtained from humans. A graph of the knee data from the CAL3D report and the prototype polynomial fit are given in figure 4.3.

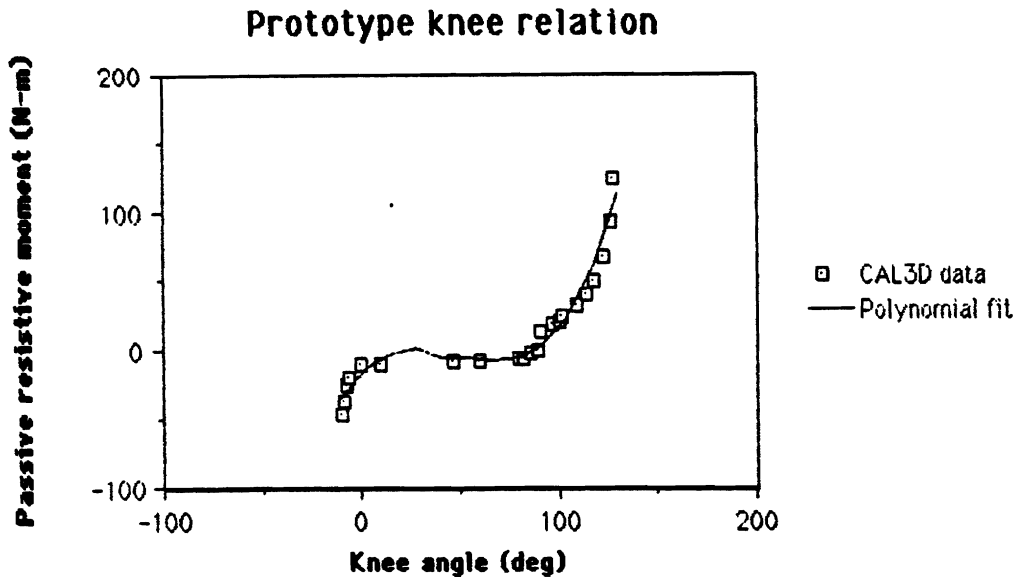


Figure 4.3: Knee moment-displacement relationship: CAL3D data and prototype knee model

### 4.2.3 Shoulder Springs

The shoulder of the prototype figure is modeled as a spherical joint. Its stiffness behavior is based on a study by Engin [20]. This study measured the passive resistive shoulder moments of several healthy subjects. Moment components about each of the coordinate axes (see figure 5.5 for definition of axes) were measured as the subjects arm was forced through its range of motion in each of several directions. As might be expected, the resistive moment tended to directly oppose the motion although small components in other directions were also measured. That is, when the arm was forced through its range of motion about the x axis, the x component of the resistive moment was the greatest. The same was true for the other axes as well.

For the prototype shoulder, the moment-displacement relationship used for each degree of freedom is a cubic polynomial fit of the moment component directly opposing the motion of the arm. The small components about the other axes are ignored.



Figures 4.4, 4.5 , and 4.6 compare the cubic polynomial fits to the data from a particular subject in Engin's study.

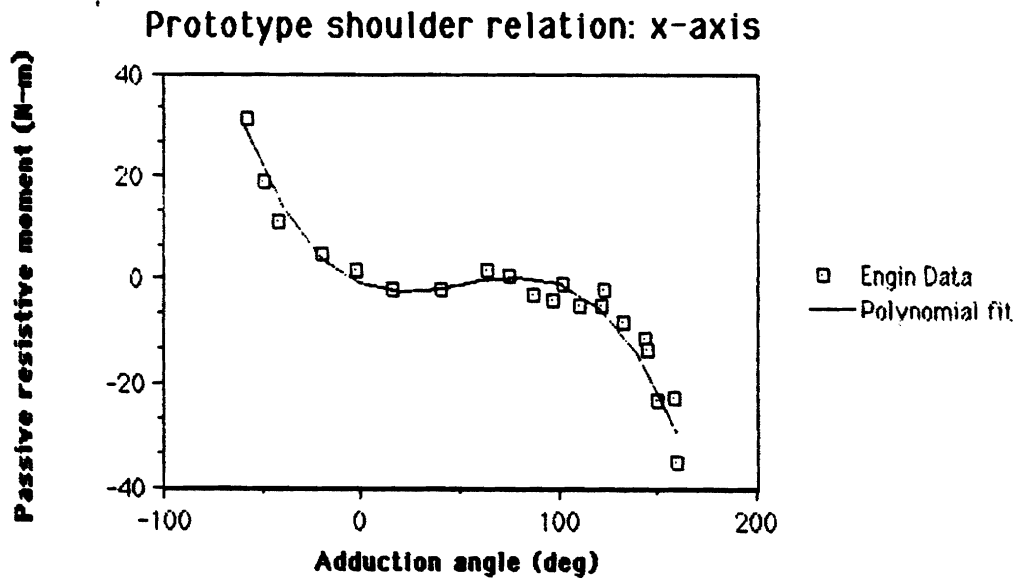


Figure 4.4: Shoulder moment-displacement relationship for x axis: Engin subject data and prototype shoulder model

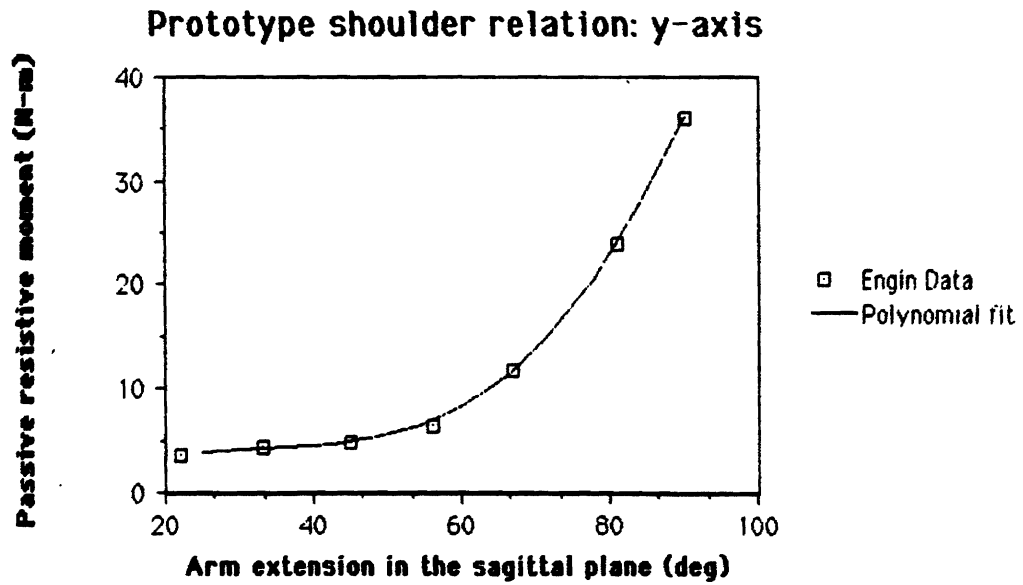


Figure 4.5: Shoulder moment-displacement relationship for y axis: Engin subject data and prototype shoulder model

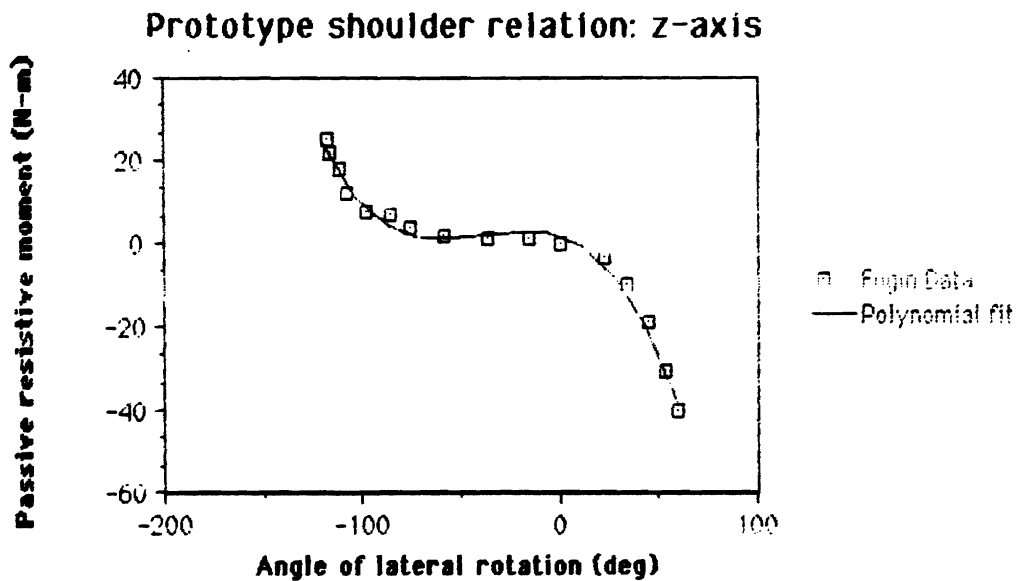


Figure 4.6: Shoulder moment-displacement relationship for z axis: Engin subject data and prototype shoulder model

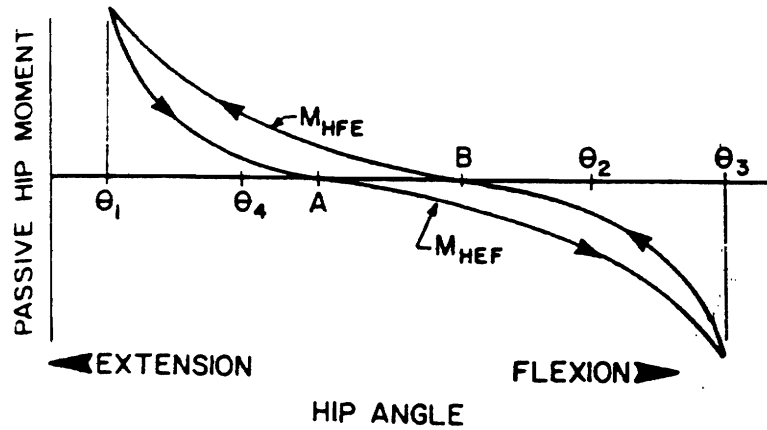


Figure 4.7: Illustration of the directional dependence of the passive hip moment.  $M_{HFE}$  is the moment for leg motion in the direction of increasing extension and  $M_{HEF}$  is the moment for increasing flexion.

#### 4.2.4 Hip Springs

The study used as a basis for the prototype hip is by Yoon and Mansour [23]. It examines motion of the leg parallel to the sagittal plane for several different knee angles. Some of the muscles involved in hip motion span both the hip and knee joints, so knee angle can strongly influence the range of motion of the hip (and vice versa). Yoon and Mansour found a qualitative relationship between knee angle and passive hip moment<sup>3</sup> but were unable to establish a good quantitative relationship.

Another phenomenon described by Yoon and Mansour is the directional dependence of the hip moment. That is, hip moments for a given displacement depended on whether the leg was moving in a direction of increasing flexion or increasing extension (see figure 4.7, adopted from Yoon and Mansour, for an illustration).

The prototype hip models neither the relationship between the hip moment and the knee angle nor the different functions for movement in flexion and movement in extension. Of these two shortcomings, the lack of a good two joint model for the hip is probably more serious. The difference between the increasing extension and

<sup>3</sup>The greatest moment in hip extension and the least moment in hip flexion occur with maximum knee flexion. And the least moment in hip extension and the greatest moment in hip flexion occur with the knee at maximum extension

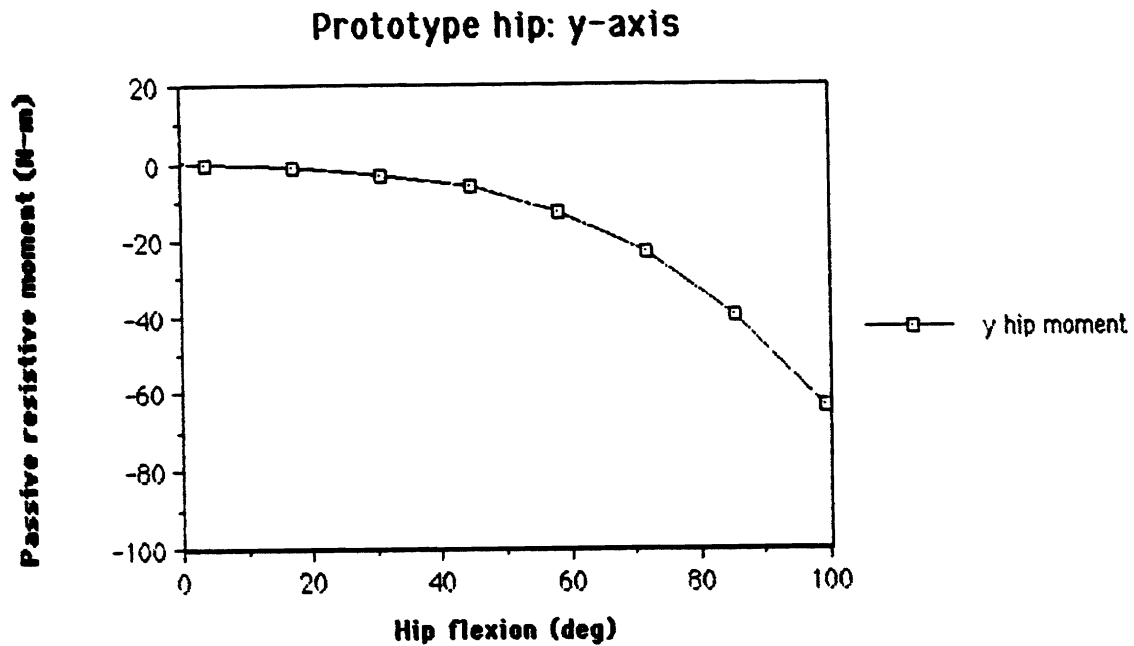


Figure 4.8: Hip moment-displacement relationship for flexion: Yoon and Mansour subject data and prototype hip model.

the increasing flexion curves was on the order of 5-10 N-m or about 15% of the total range. The difference between hip moments of a bent leg and a straight leg is much greater, as much as 45-50 N-m or about 60% of the total range.

It is reasonable to compensate for the directional dependence property by fitting a function that lies on the median between the increasing extension and the increasing flexion curves. This approximation would result in values that deviated from the observed values by less than 8%.

Without a reasonable two joint function for the hip moment, though, it is very difficult to account for the dependence on knee angle.

The prototype hip uses the data described by Yoon and Mansour for a particular subject with knee angle of 15 degrees and movement in the direction of increasing extension. This particular scenario was chosen arbitrarily. Figure 4.8 compares the Yoon and Mansour data to the prototype hip model. Moment values for the other degrees of freedom (ab/adduction and medial/lateral rotation) are estimates not based on any empirical study. Figures 4.9 and 4.10 show the relationships used.

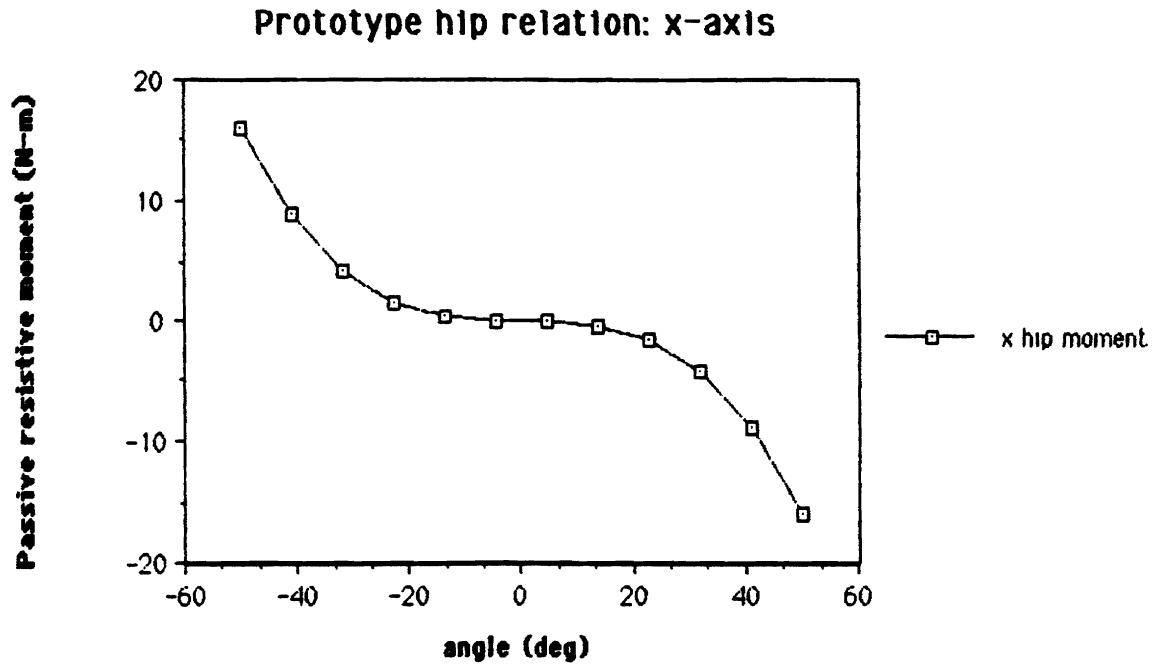


Figure 4.9: Prototype hip moment-displacement relationship for ab/adduction.

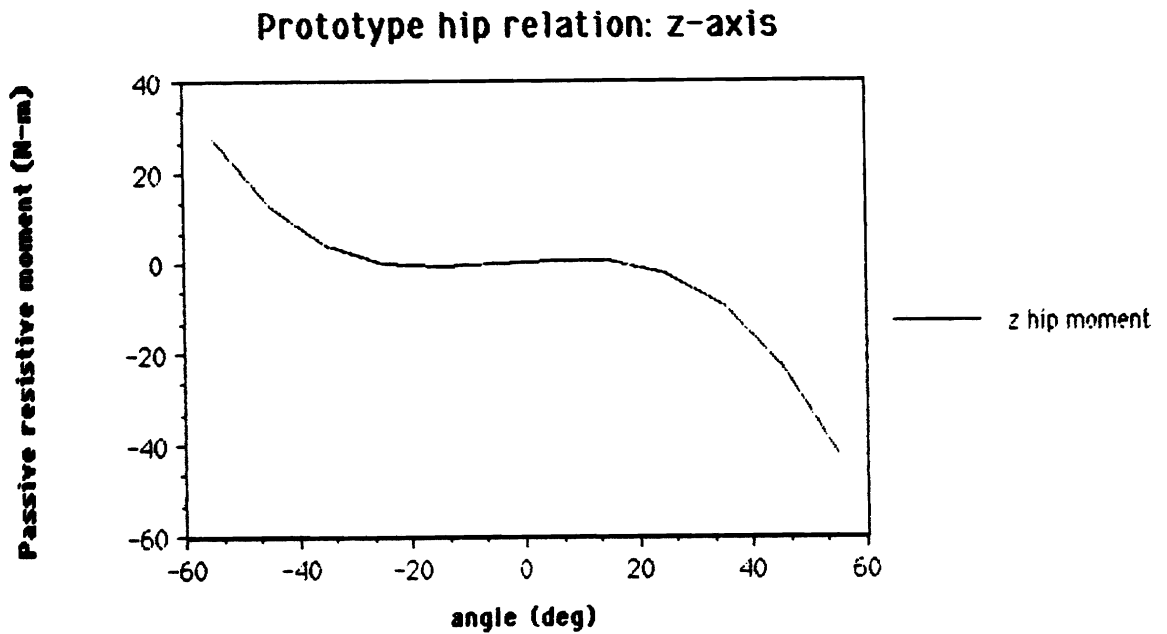


Figure 4.10: Prototype hip moment-displacement relationship for medial/lateral rotation.

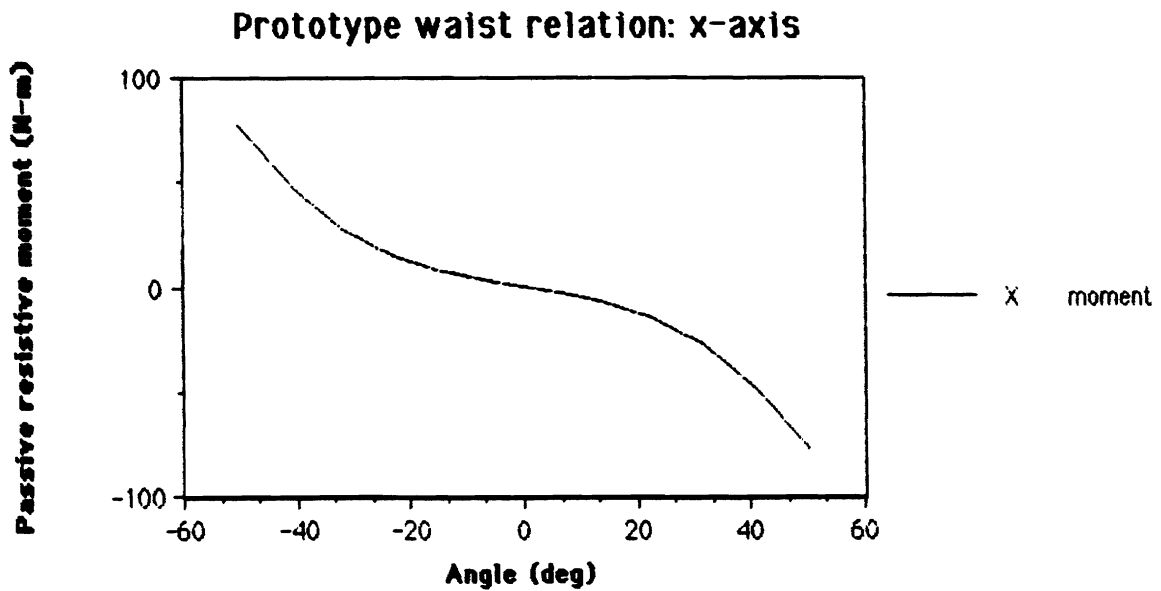


Figure 4.11: Prototype waist moment-displacement relationship for flexion parallel to the frontal plane.

#### 4.2.5 Waist Springs

The waist joint in the prototype figure is an artifice. No such joint exists in the human figure. The waist joint is simply a device to allow some bending of the torso without adding the complexity of a curvable spine. The moment-displacement functions used, therefore, are completely artificial. Graphs of the relations used are given in figures 4.11, 4.12, and 4.13.

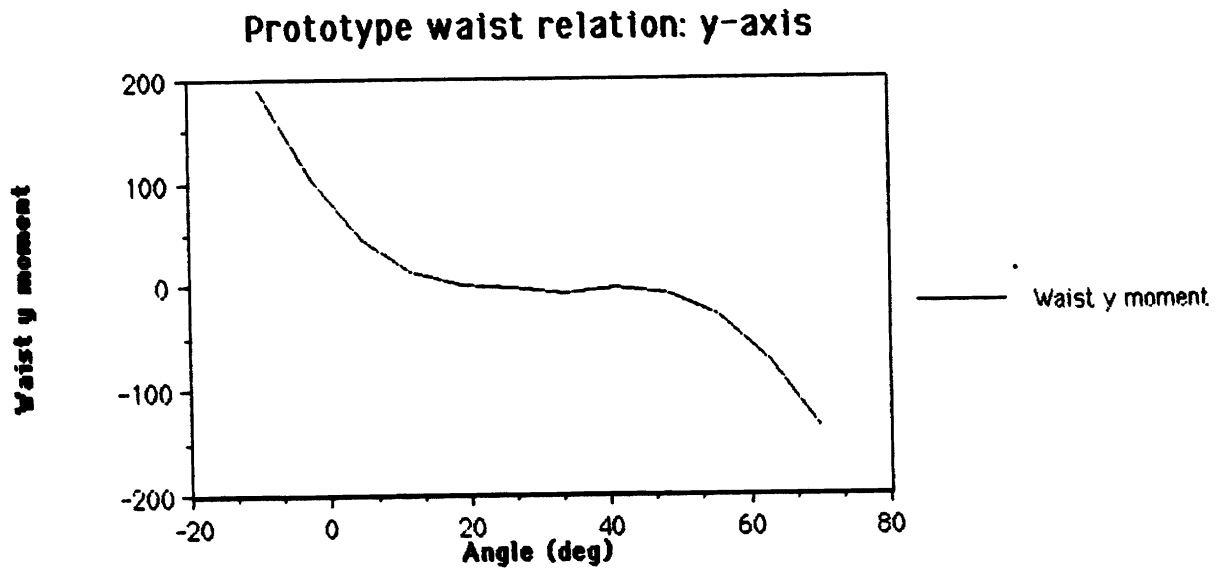


Figure 4.12: Prototype waist moment-displacement relationship for flexion parallel to the sagittal plane.

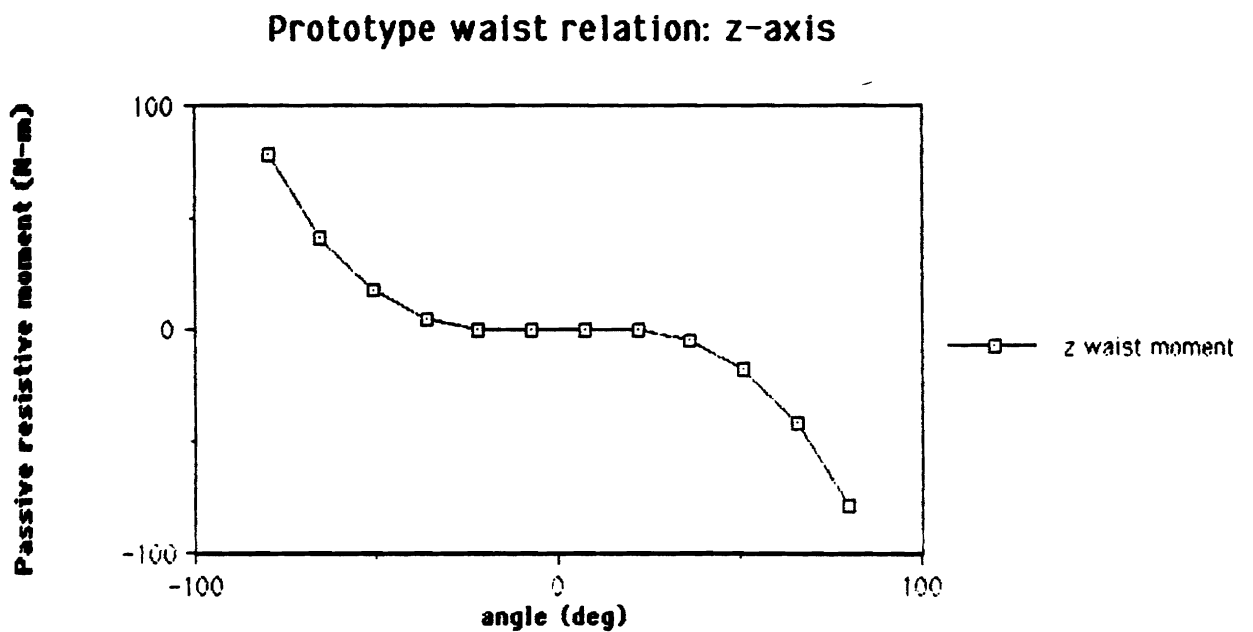


Figure 4.13: Prototype waist moment-displacement relationship for rotation parallel to the transverse plane.

# Chapter 5

## Positioning Tools

Computer animation can be created in two ways. It may be generated by algorithm (with a simulation program, for example), or it can be crafted keyframe by keyframe by an animator. In the latter case, it is essential that the animator have at his or her disposal tools that will help position the figure within each keyframe. The human figure, with its many joints presents a particular challenge for the keyframe animator as each degree of freedom of each joint must be correctly positioned if the resulting motion is to seem natural. In this chapter, I describe some of the positioning tools I developed to aid the animator in creating more natural body positions. Most of the routines have a more general range of application, though. The center of mass routines and the geometry-based mass and inertia information generator are general routines that could be applied to any Peabody figure. The multiple joint positioning scheme can only be applied to joints with revolute degrees of freedom, but otherwise, it is independent of the choice of figure. Only the coupled shoulder routine requires the specific human figure to function properly.

### 5.1 Locating the Center of Mass

Two positioning tools provide the animator with the location of the center of mass of a figure. This information is clearly useful in creating some kinds of motions (jumping



and other whole body motions, for example) as well as for the analysis of motions already created by simulation programs.

The center of mass of an articulated figure changes as the figure moves. These tools compute the center of mass of the figure and display it either interactively, as joints are adjusted and the mass distribution of the figure changes, or as a separate calculation on a static figure. Both routines make use of the special “CM” site and the “mass” field within the Peabody representation of each segment. The basic algorithm for both is based on the definition of the center of mass of a collection of bodies:

$$\mathbf{x}_{\text{CM}} = \frac{\sum_{i=1}^n \mathbf{x}_i m_i}{\sum_{i=1}^n m_i} \quad (5.1)$$

Here  $\mathbf{x}_i$  is the global position vector of segment  $i$ ,  $m_i$  is the mass of segment  $i$ , and  $\mathbf{x}_{\text{CM}}$  is the global position of the center of mass of the set of  $n$  segments composing the figure.

Both center of mass routines ask the user to identify a figure either directly, by selecting it with the mouse, or indirectly, by selecting a joint that is a part of the figure. Each routine then searches through the data structure and sums over the masses of each segment and the global positions of each site named “CM”. The equation 5.1 is applied, and the result is displayed on the screen numerically as well as visually, with a coordinate frame icon drawn at the figure’s center of mass.

## 5.2 Testing for Figure Stability

One possible application for the calculation of the center of mass is to determine static support stability of a figure. The simplest case is treated by the “Stability Test” menu choice in the Center of Mass menu of my application program. This routine asks the user to select a figure and a set of supporting faces. The program then calculates the center of mass of the whole figure and determines whether or not

the projection of the center of mass onto the ground plane falls within a support polygon defined by the convex hull of the projections of the vertices of the supporting faces onto the ground plane. Figure 5.1 illustrates the method. A message is returned by the program indicating whether or not the figure is balanced.

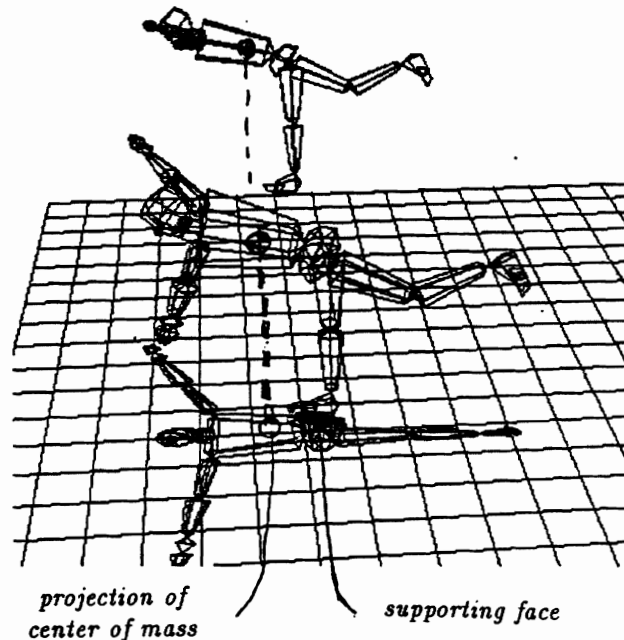


Figure 5.1: Determining Static Figure Stability

This stability test is a simple example of how the center of mass routine might be used by an application program. The test by itself may be somewhat useful when considering the posture of a human figure carrying a massive load and could be used iteratively as the posture was adjusted from an unbalanced one to a more stable one.

The test assumes that the only supports available are the ones indicated by the user and that the only force acting on the figure is gravity. Further, the test assumes all joints in the figure are rigid.

### 5.3 Generating Mass and Inertia Data

The routines that calculate the figure center of mass and the stability test both depend on the existence of the special "CM" site. The "inertia" field of the Peabody figure

description is similarly assumed to exist by the program that converts the figure to DYSPAM format. This information typically comes from the user, but on occasion it would be useful if such data could be estimated automatically. For this purpose, a special routine is included as a choice in the Center of Mass menu.

The routine, identified as “Psurf Mass Info” asks the user to pick a segment with the mouse and enter its density. It then displays on the screen the center of mass location, volume, and moment of inertia matrix based on an assumption of uniform density. Such an assumption is reasonable for many applications. The psurf selected may be of any shape, concave or convex. The only restriction is that the vertices <sup>1</sup>of each face be ordered in such a way that an observer on the exterior of the solid would see them sequentially arranged in a counter-clockwise direction. This restriction is already required by various rendering algorithms

The algorithm for determining mass properties is based on the idea that any solid with planar faces can be systematically decomposed into a set of tetrahedra. One method for achieving this decomposition is selecting one vertex as the apex <sup>2</sup>of all the tetrahedra, and then systematically dividing each face into triangles with each triangle serving as the base of a tetrahedron. Figure 5.2 illustrates how a rectangular prism might be decomposed. In the scheme I use, some of the tetrahedra are degenerate, having all four points coplanar. This case is detected by the program and does not present a problem. Once the decomposition is accomplished, the task of finding mass properties of the whole solid is reduced to the two tasks of finding the mass properties of each tetrahedron and using those values to calculate the properties of the whole solid.

The first property to consider is volume. In a solid of uniform density the volume is

---

<sup>1</sup>In this discussion, “vertex” will always refer to a point on the surface of the psurf that is serving to define the boundary of a face.

<sup>2</sup>“Apex” is the name assigned arbitrarily to one of the 4 corners of a tetrahedron. “Base” is the set of three corners that are not the apex.

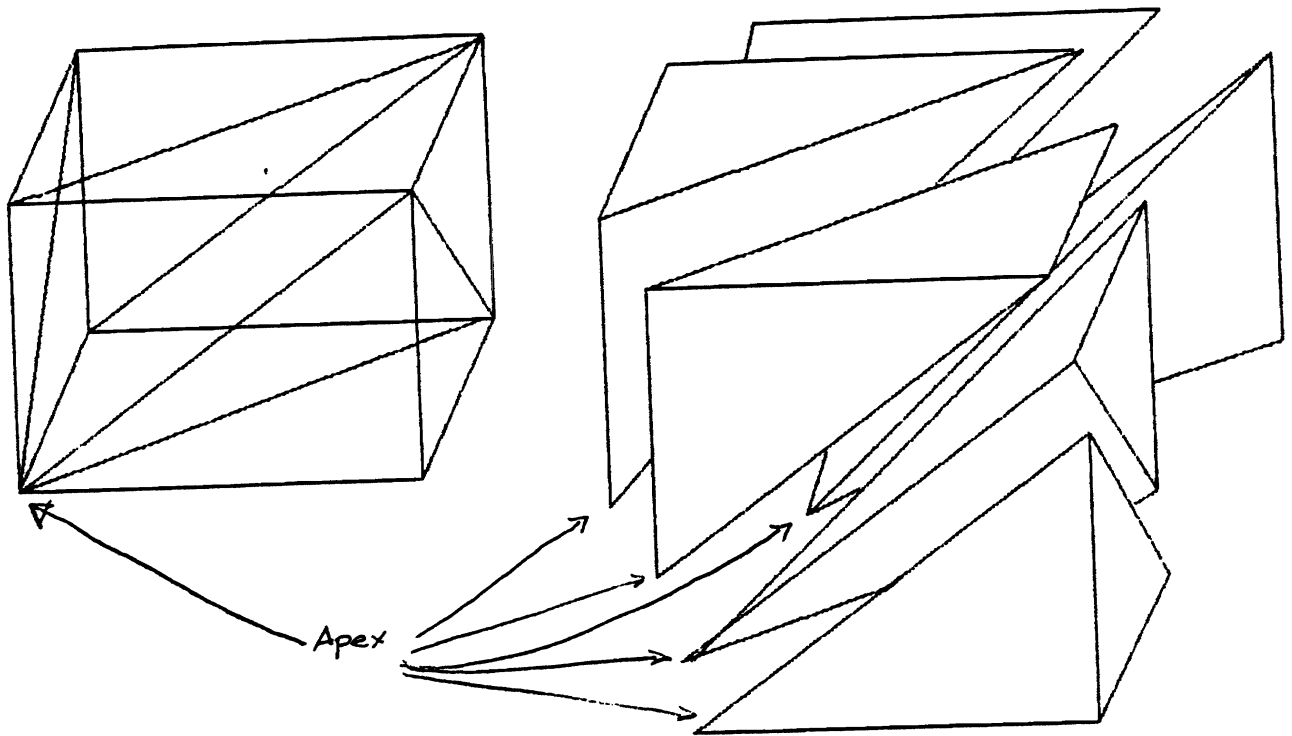


Figure 5.2: Example of decomposing a solid into tetrahedra

proportional to the mass, so if the density is known, determining the volume effectively determines the mass. In the calculations that follow, the density constant is omitted with understanding that the results should be multiplied by the density to change volume properties into mass properties.

The volume of a tetrahedron may be calculated in two ways, via vector products, or by direct integration. The vector method uses the formula:

$$\text{Volume} = \frac{\mathbf{a}_1 \times \mathbf{a}_2 \cdot \mathbf{a}_3}{6} \quad (5.2)$$

In this formula the three vectors  $\mathbf{a}_1$ ,  $\mathbf{a}_2$ , and  $\mathbf{a}_3$  originate at the apex of the tetrahedron and extend to each of the vertices in the base in order. Notice that since this formula contains a vector cross product, the order of the vectors is important. If the vertices of a face are arranged in a clockwise direction from the vantage point of the apex, then by the convention described earlier, the apex of the tetrahedron must be on the

interior side of that face. Similarly, if the vertices are arranged counter-clockwise, the apex is on the exterior. The volume value calculated is positive when the apex sees the inside of a face and negative when the apex sees the outside. If the apex is itself a vertex (as in my implementation), all the volumes calculated for a convex solid will be either zero (for faces including the apex) or some positive value. Concave solids or solids not simply connected will produce some positive and some negative volumes. The sum of these volumes is the total volume of the solid.

The other method for finding the volume of an arbitrary tetrahedron, direct integration, is not immediately necessary since the vector formula (equation 5.2) is available. However, the method of finding a volume integral for an arbitrary tetrahedron will be necessary when calculating the elements of the inertia tensor. For illustration, I will provide the method in the following discussion.

The difficulty in calculating a volume integral over an arbitrary tetrahedron is in the setting of the limits of integration. Since the shape may be positioned anywhere in space and oriented in any way, it is difficult to arrive at a general algorithm that does not rely on a classification scheme with many cases. A better solution is to find a way to transform any tetrahedron into a special case where the limits are easy to set and the integration is straightforward. In my implementation, this simple case is one where the apex of the tetrahedron is at the origin of a coordinate system and each of the base vertices lies on a coordinate axis. In this case the volume integral is clearly given by:

$$\text{Volume} = \int_0^t \int_0^{s(1-z/t)} \int_0^{r(1-y/s-z/t)} dx dy dz \quad (5.3)$$

Here,  $r$ ,  $s$ , and  $t$  are the  $x$ ,  $y$  and  $z$  intercepts, respectively. Figure 5.3 illustrates the definition of these variables. The solution is easily found to be:

$$\text{Volume} = \frac{rst}{6} \quad (5.4)$$

It is obvious that for this special case of a tetrahedron with three orthogonal edges, this result is equivalent to the result given by the vector equation stated previously.

To extend this result to arbitrary tetrahedra, a transformation to a (possibly) non-orthogonal coordinate system must be made. For convenience, select the three vectors  $\mathbf{a}_1$ ,  $\mathbf{a}_2$ , and  $\mathbf{a}_3$  (as defined earlier) as the basis vectors for the new system,  $\epsilon_1$ ,  $\epsilon_2$  and  $\epsilon_3$ . So we can say:

$$\epsilon_1 = \mathbf{a}_1 = a_{11}\mathbf{i} + a_{12}\mathbf{j} + a_{13}\mathbf{k} \quad (5.5)$$

$$\epsilon_2 = \mathbf{a}_2 = a_{21}\mathbf{i} + a_{22}\mathbf{j} + a_{23}\mathbf{k} \quad (5.6)$$

$$\epsilon_3 = \mathbf{a}_3 = a_{31}\mathbf{i} + a_{32}\mathbf{j} + a_{33}\mathbf{k} \quad (5.7)$$

Here,  $\mathbf{i}$ ,  $\mathbf{j}$  and  $\mathbf{k}$  are the usual Cartesian unit basis vectors.

For convenience let us create a matrix  $[\mathbf{A}]$  composed of the coefficients in the above expressions.

Notice that a single position  $\mathbf{p}$  can be expressed as a composition of scalars times the basis vectors of either system. That is,

$$\mathbf{p} = p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k} \quad (5.8)$$

Or

$$\mathbf{p} = \xi^1\epsilon_1 + \xi^2\epsilon_2 + \xi^3\epsilon_3 \quad (5.9)$$

What is needed now is a transformation from the Cartesian space to the non-orthogonal space. To find this we make use of the metric tensor  $[\mathbf{g}]$ . The metric tensor has nine elements,  $g^{i,j}$ . Here are some of its properties:

$$g^{ij} = \epsilon^i \cdot \epsilon^j \quad (5.10)$$

$$g_{ij} = \epsilon_i \cdot \epsilon_j \quad (5.11)$$

$$g^{ip}g_{pj} = \delta_j^i \quad (5.12)$$

$$\xi^i = g^{ij} \xi_j \quad (5.13)$$

In these equations, the  $\epsilon^i$  are the contravariant basis vectors. They are defined to be orthogonal to the covariant basis vectors  $\epsilon_i$ .  $\delta^i_j$  is the Kronecker delta.

The covariant components in the non-orthogonal system  $\xi_j$  on the right hand side of equation 5.13 are defined as the projections of some vector  $\mathbf{p}$  on the covariant basis vectors:

$$\xi_j = \mathbf{p} \cdot \epsilon_j \quad (5.14)$$

So from Eqs. 5.14, 5.13 and the definitions of the non-orthogonal basis vectors  $\epsilon$  (equations 5.5–5.7) , we can write:

$$\xi^i = g^{ij} \mathbf{p} \cdot \mathbf{a}_j \quad (5.15)$$

If we call the metric tensor  $[\mathbf{g}]$  and we recall the coefficient matrix  $[\mathbf{A}]$ , we rewrite equation 5.15 in matrix form:

$$\begin{bmatrix} \xi^1 \\ \xi^2 \\ \xi^3 \end{bmatrix} = [\mathbf{g}][\mathbf{A}] \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \quad (5.16)$$

Here,  $p_1, p_2, p_3$  are the Cartesian components of  $\mathbf{p}$ .

It can be shown that  $([\mathbf{g}][\mathbf{A}])^{-1}$  is just  $[\mathbf{A}]^T$ . So a transformation from the non-orthogonal system to the Cartesian system can also be written:

$$\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = [\mathbf{A}]^T \begin{bmatrix} \xi^1 \\ \xi^2 \\ \xi^3 \end{bmatrix} \quad (5.17)$$

The differential volume element of this system is given by: <sup>3</sup>

$$dV = dx dy dz = J d\xi^1 d\xi^2 d\xi^3 = \sqrt{\text{Det}[\mathbf{g}]} d\xi^1 d\xi^2 d\xi^3 \quad (5.18)$$

The next step in transforming the volume integral is determining the limits of integration. The limits are portions of the equation of a plane in intercept form. So

---

<sup>3</sup>For a full development of this result see Budiansky [36]

transforming the limits is equivalent to transforming that plane. In Cartesian space, the base plane of the tetrahedron is defined by the three points located by the vectors  $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$  and the equation may be given by:

$$b_1 p_1 + b_2 p_2 + b_3 p_3 + b_4 = 0 \quad (5.19)$$

Here, the  $b$ 's are constants.

The limits of integration in the non-orthogonal system are found by making the substitution for the  $p$ 's given in 5.17 resulting in the transformed equation for the base plane:

$$(b_1 a_{11} + b_2 a_{12} + b_3 a_{13})\xi^1 + (b_1 a_{21} + b_2 a_{22} + b_3 a_{23})\xi^2 + (b_1 a_{31} + b_2 a_{32} + b_3 a_{33})\xi^3 + b_4 = 0 \quad (5.20)$$

A final manipulation places this equation in intercept form:

$$\frac{\xi^1}{r} + \frac{\xi^2}{s} + \frac{\xi^3}{t} = 1 \quad (5.21)$$

$r, s$  and  $t$  are the transformed intercept values and can easily be found from equation 5.20. Finally, the integral given in equation 5.3 can be written and calculated for the non-orthogonal case:

$$\text{Volume} = \sqrt{\text{Det}[\mathbf{g}]} \int_0^t \int_0^{s(1-\xi^3/t)} \int_0^{r(1-\xi^2/s-\xi^3/t)} d\xi^1 d\xi^2 d\xi^3 \quad (5.22)$$

$$\text{Volume} = \sqrt{\text{Det}[\mathbf{g}]} \frac{rst}{6} \quad (5.23)$$

The same approach can be used to calculate center of mass and moments and products of inertia of arbitrary tetrahedra. In the case of the  $x$  location of center of mass, the Cartesian volume integral (for the special case) is :

$$\bar{x} = \int_0^t \int_0^{s(1-z/t)} \int_0^{r(1-y/s-z/t)} x dx dy dz \quad (5.24)$$

The  $\bar{y}$  and  $\bar{z}$  equations are similar. To determine the  $x$  location of the center of mass for an arbitrary tetrahedron, the limits are the same as for the volume calculation,



but the integrand must be transformed. This transformation is very straight forward and yields:

$$\bar{x} = \sqrt{\text{Det}[\mathbf{g}]} \int_0^t \int_0^{s(1-\xi^3/t)} \int_0^{r(1-\xi^2/s-\xi^3/t)} (a_{11}\xi^1 + a_{21}\xi^2 + a_{31}\xi^3) d\xi^1 d\xi^2 d\xi^3 \quad (5.25)$$

$$\bar{x} = \sqrt{\text{Det}[\mathbf{g}]} \frac{a_{11}r^2st + a_{21}rs^2t + a_{31}rst^2}{24\text{Volume}} \quad (5.26)$$

And of course, the other coordinates of the center of mass can be found in the same way. This location is the center of mass relative to the apex (origin of the non-orthogonal coordinate system). Since in my implementation, all of the tetrahedra composing the solid have the same apex, finding the center of mass of the whole solid is simply a matter of performing a mass (or volume) weighted average over the collection of tetrahedra and then adjusting the result to compensate for the displacement of the apex vertex from the origin. Continuing the calculation for  $\bar{x}$ :

$$\bar{x}_{\text{total}} = (\text{x coord of apex}) + \frac{\sum_{i=1}^n \bar{x}_i \text{Volume}_i}{\text{Total Volume}} \quad (5.27)$$

Elements of the inertia tensor can be found in the same way. The evaluation of the integrals in the transformed space, while straight forward, are extremely lengthy. The symbolic math program MACSYMA was used to both verify hand calculations and to generate evaluations of these integrals. Basically, the inertia tensor is composed of two kinds of terms - diagonal terms, and off-diagonal terms. A sample of an initial integral and its solution for each kind is given below.

The inertia tensor in its usual form is:

$$\begin{bmatrix} \int \int \int (y^2 + z^2) dx dy dz & - \int \int \int xy dx dy dz & - \int \int \int xz dx dy dz \\ - \int \int \int yx dx dy dz & \int \int \int (x^2 + z^2) dx dy dz & - \int \int \int yz dx dy dz \\ - \int \int \int zx dx dy dz & - \int \int \int zy dx dy dz & \int \int \int (x^2 + y^2) dx dy dz \end{bmatrix} \quad (5.28)$$

A typical diagonal term (the first element (1,1)) yields:

$$I_x = \int \int \int (y^2 + z^2) dx dy dz \quad (5.29)$$

$$\begin{aligned}
I_x = & \sqrt{\text{Det}[\mathbf{g}]} \int_0^t \int_0^{s(1-\xi^3/t)} \int_0^{r(1-\xi^2/s-\xi^3/t)} (a_{12}\xi^1 + a_{22}\xi^2 + a_{32}\xi^3)^2 \\
& + (a_{13}\xi^1 + a_{23}\xi^2 + a_{33}\xi^3)^2 d\xi^1 d\xi^2 d\xi^3
\end{aligned} \tag{5.30}$$

$$\begin{aligned}
I_x = & \sqrt{\text{Det}[\mathbf{g}]} \left[ (a_{12}^2 + a_{13}^2)r^3st + (a_{22}^2 + a_{23}^2)rs^3t + (a_{32}^2 + a_{33}^2)rst^3 \right. \\
& + (a_{12}a_{22} + a_{13}a_{23})r^2s^2t + (a_{12}a_{32} + a_{13}a_{33})r^2st^2 \\
& \left. + (a_{22}a_{32} + a_{23}a_{33})rs^2t^2 \right] / 60
\end{aligned} \tag{5.31}$$

A typical off-diagonal element (element (2,1)) is given by:

$$I_{yx} = \int \int \int yx \, dx dy dz \tag{5.32}$$

$$\begin{aligned}
I_{yx} = & \sqrt{\text{Det}[\mathbf{g}]} \int_0^t \int_0^{s(1-\xi^3/t)} \int_0^{r(1-\xi^2/s-\xi^3/t)} (a_{12}\xi^1 + a_{22}\xi^2 + a_{32}\xi^3) \\
& (a_{11}\xi^1 + a_{21}\xi^2 + a_{31}\xi^3) d\xi^1 d\xi^2 d\xi^3
\end{aligned} \tag{5.33}$$

$$\begin{aligned}
I_{yx} = & \sqrt{\text{Det}[\mathbf{g}]} \left[ (a_{12}a_{11}r^3st + a_{22}a_{21}rs^3t + a_{32}a_{31}rst^3) / 60 + [(a_{12}a_{21} + a_{22}a_{11})r^2s^2t \right. \\
& \left. + (a_{12}a_{31} + a_{32}a_{11})r^2st^2 + (a_{22}a_{31} + a_{32}a_{21})rs^2t^2 \right] / 120
\end{aligned} \tag{5.34}$$

The results for each tetrahedron are relative to the local origin (i.e. the apex vertex). To find the results relative to the center of mass of the solid, we first find the total inertial terms relative to the apex by adding up the contributions of each tetrahedron:

$$\tilde{I}_x = \sum_{i=1}^n (I_x)_i \tag{5.35}$$

$$\tilde{I}_{yx} = \sum_{i=1}^n (I_{yx})_i \tag{5.36}$$

The  $\tilde{I}$  are elements of the inertia matrix for the whole solid but relative to the apex vertex.

The next step is to apply the parallel axis theorem to find the corresponding inertia values relative to the center of mass. For diagonal terms the formula is:

$$\bar{I}_x = \tilde{I}_x - (\text{Total volume})(\tilde{y}^2 + \tilde{z}^2) \quad (5.37)$$

And for the off-diagonal terms the formula is:

$$\bar{I}_{yx} = \tilde{I}_{yx} - (\text{Total volume})(\tilde{y}\tilde{x}) \quad (5.38)$$

Where the  $\tilde{x}, \tilde{y}$  and  $\tilde{z}$  indicate the location of the center of mass relative to the apex vertex.

## 5.4 Parametric Shoulder Positioning

The shoulder “joint” is actually a system of several articulations stabilized and controlled by 13 muscles and 3 major bones. Although our model simplifies this complex system into a chain of two segments (clavicle and humerus) with 5 degrees of freedom (3 at the glenohumeral “joint” and 2 at the sternoclavicular “joint”) specifying the position of the upper arm and clavicle relative to the torso remains a particularly challenging task. It is inconvenient to specify 5 degrees of freedom merely to position the arm interactively, and with the user free to chose any values for the 5 angles, many unfeasible arm positions may result.

To assist the user in positioning the arm, I developed an interactive positioning scheme in which the user specifies, in spherical coordinates, the position of a point on the humerus, and the program continuously calculates and sets appropriate values for clavicle elevation and abduction and humerus flexion/extension, ab/adduction and medial-lateral rotation. In this way, the user can easily achieve approximately the shoulder configuration that he seeks with a single command.

Such a routine depends on the availabilty of simple relationships between the 5 degrees of freedom being set and the two that the user specifies. Inman [20] provides

a thorough description of the anatomy of the shoulder complex and the relationships between the various structures that participate in shoulder motion. In particular, Inman provides a graph based on clinical observations showing the relationship between humerus elevation (both in abduction and forward flexion) and clavicle elevation. A reproduction of this graph appears in figure 5.4. Notice that arm elevation is not equivalent to humerus elevation since humerus elevation is modeled as relative to the clavicle and not the torso. This suggests that arm elevation (defined relative to the torso) is actually the sum of contributions from both the sternoclavicular joint and the glenohumeral joint. The graph in figure 5.4 provides the sternoclavicular contribution for a given elevation, so the contribution at the glenohumeral joint is the difference.

The shoulder positioning routine uses the two values representing the “latitude”  $\phi$  and the “longitude”  $\theta$  of the elbow to determine the orientations of the clavicle and humerus.  $\theta$  is zero with the arm pointing straight in front of the figure and increases as the arm moves to the figure’s left.  $\phi$  is zero when the arm is pointing straight overhead and increases as the arm is lowered (this is opposite to the definition of increasing arm elevation). With these definitions established, we can now define the formula relating  $\phi$  and  $\theta$  to the various joint angles.

For the left shoulder:

$$\text{elevation angle} = \alpha_{el} = 180 - \phi \quad (5.39)$$

$$\text{abduction angle} = \alpha_{ab} = 90 - \theta \quad (5.40)$$

$$\text{clavicle elevation due to motion parallel to frontal plane} = \beta_1 \quad (5.41)$$

$$\beta_1 = \begin{cases} 0.2514\alpha_{el} + 91.076 & \text{for } 0 \leq \alpha_{el} \leq 131.4 \\ -0.035\alpha_{el} + 128.7 & \text{for } \alpha_{el} > 131.4 \end{cases} \quad (5.42)$$

$$\text{clavicle elevation due to motion parallel to sagittal plane} = \beta_2 \quad (5.43)$$

$$\beta_2 = \begin{cases} 0.21066\alpha_{el} + 92.348 & \text{for } 0 \leq \alpha_{el} \leq 130.0 \\ 120.0 & \text{for } \alpha_{el} > 130.0 \end{cases} \quad (5.44)$$

$$\text{clavicle angle 1} = \cos(\alpha_{ab})\beta_1 + (1 - \cos(\alpha_{ab}))\beta_2 - 90 \quad (5.45)$$

$$\text{clavicle angle 2} = 0.2\alpha_{ab} \quad (5.46)$$

$$\text{humerus angle 1} = \alpha_{e1} - \text{clavicle angle 1} \quad (5.47)$$

$$\text{humerus angle 2} = \alpha_{ab} - \text{clavicle angle 2} \quad (5.48)$$

In these equations, the  $\alpha$ 's are upper arm angles as measured relative to the torso, and the 2  $\beta$ 's are "clavicle" angles as defined in Inman's graph (figure 5.4). The joint angles themselves are defined as shown in figure 5.5. All the equations use degrees as the measure of the angles.

Equations 5.42 and 5.44 were adopted from a linear approximation of Inman's graph. The other clavicle angle relation was an estimate. It is important to note that even Inman's data should be considered merely a representative case rather than a universally valid relationship. Inman was concerned with qualitative observations more than quantitative relationships.

The above formulae establish 4 of the 5 joint displacements required to uniquely position the upper arm system. The fifth displacement is medial-lateral rotation of the humerus, or rotation about the long axis of the upper arm. This angle of rotation or "twist" could be determined in any of a number of ways. The simplest would be to let the angle of twist be zero. That is, whatever medial-lateral rotation was the result of the sequential rotations in abduction and elevation would be considered the default. This selection of a default angle of twist causes some difficulty at the poles of the sphere of motion, though. When the arm is pointing straight up, the twist could have a variety of different default values. In this case, the twist would depend on the path the arm to arrive at the pole, rather than being some fixed certain value at the pole regardless of the route used to arrive there.

A perhaps more restrictive scheme would be to establish a standard (or default) twist angle for every point in the sphere of motion regardless of how the arm arrived

there. Such a scheme is used in Labanotation (a variety of dance notation) and was suggested by Badler, O'Rourke and Kaufman [37] for this particular positioning problem.

This scheme is best described by simply listing the equations that define it. Below,  $\phi$  and  $\theta$  are the coordinates described earlier, and  $\psi$  is the *additional* twist imposed on the humerus after rotations of  $\theta$  and then  $\phi$  have been applied in sequence about their appropriate axes. Looking down the arm, the rotation angle  $\psi$  increases in a clockwise direction for the right arm and a counter clockwise direction for the left arm. For the left arm, the formulae are:

$$\psi = \begin{cases} 180 - \theta(1 - \frac{\phi}{90}) & -90 \leq \theta \leq 90 & 0 \leq \phi \leq 90 \\ -180(1 - \frac{\phi}{90}) + (\phi - 270)(\frac{\theta}{90}) & -90 \leq \theta \leq 90 & 90 \leq \phi \leq 180 \\ (\theta - 180)(1 + \frac{180-\phi}{90}) & 90 \leq \theta \leq 270 & 90 \leq \phi \leq 180 \\ 2(\theta - 180)\frac{\phi}{90} - (180 + \theta)(1 - \frac{\phi}{90}) & 90 \leq \theta \leq 180 & 0 \leq \phi \leq 90 \\ (180 - \theta)(1 - \frac{3\phi}{90}) + \text{sgn}(225 - \theta)360\frac{\phi}{90} & 180 \leq \theta \leq 270 & 0 \leq \phi \leq 90 \end{cases} \quad (5.49)$$

The twist angle in this scheme varies continuously over the whole sphere of motion except for a seam that occurs behind the figure's back in a physically unreachable region. This scheme can be thought of as producing a "natural" rotation for the arm throughout its reachable space. This would be the rotation you would choose without thinking if instructed to point your hand in some direction. When the hand is directly overhead, the twist is such that the thumb (with no wrist or forearm rotation) would point behind the figure. When the hand is straight ahead of the figure, the thumb would point straight up. Of course this choice of preferred angle of twist is by no means unique, but with its use the user can be assured of remaining within the range of physically viable arm configurations.

In this discussion, all the formulae have been expressed for the left shoulder and its joint angle conventions. The formulae, are, of course, about the same for the right arm with the differences mostly being a matter of sign conventions.

This shoulder positioning tool provides a new level of detail to the motion of the

shoulder without further encumbering the user. The parametrically related shoulder joints reflect the moving center of rotation that is an important characteristic of the human shoulder. This would be lost if a simple ball joint is used as the model.

## 5.5 Multiple Interactive Joint Positioning

The idea of linking the motion of one joint or one degree of freedom to another can be expanded to facilitate the positioning of whole sets of joints if a specific relationship between their joint angles is known. A simple application of this idea is symmetrical arm or leg motion. If it is known that both elbows will always have the same or nearly the same angle, it should not be necessary to set each of them separately. Instead, the user would specify the relationship between the two joints and then adjust one of them with the angle of the other being set automatically.

The interactive linked motion routine I developed allows the user to specify other joints as being proportionally related to a single controlling joint. This will allow various kinds of symmetric and anti-symmetric relationships to be created. The user specifies the number of joints to be linked to the controlling joint and the factor  $c$  that will be the multiplier for the angles of that joint. The relationships for a joint  $i$  with three degrees of freedom can be stated:

$$\begin{aligned}\theta_{i_0} &= c_i \theta_{\text{control}_0} \\ \theta_{i_1} &= c_i \theta_{\text{control}_1} \\ \theta_{i_2} &= c_i \theta_{\text{control}_2}\end{aligned}\tag{5.50}$$

Here,  $c$  is a multiplier specified by the user,  $\theta_{i_0}$  is the zeroth joint angle of joint  $i$ , and  $\theta_{\text{control}_0}$  is the zeroth joint angle of the designated controlling joint. If the constant  $c$  is chosen to be 1.0, then the dependent joint will be set so that it has exactly the same joint angles as the controlling joint. Up to 10 dependent joints can be chosen. Also, the values of the constants are stored in static memory, so the next time the routine is invoked, the same relationship may be used without having to specify it again.

This algorithm is a little primitive as the parameterization really should be by degree of freedom rather than by joint. This is particularly apparent when one attempts to use the routine on joints with dissimilar numbers of degrees of freedom, or on joints that are symmetrically positioned within the figure (like left and right shoulders) but whose degrees of freedom are not defined symmetrically.

Nevertheless, the concept of a user specified relationship between joints that can be modified and used interactively is a powerful one. One could easily imagine how this idea could be extended to affect the geometry of the psurf or any other values in the graphics environment.



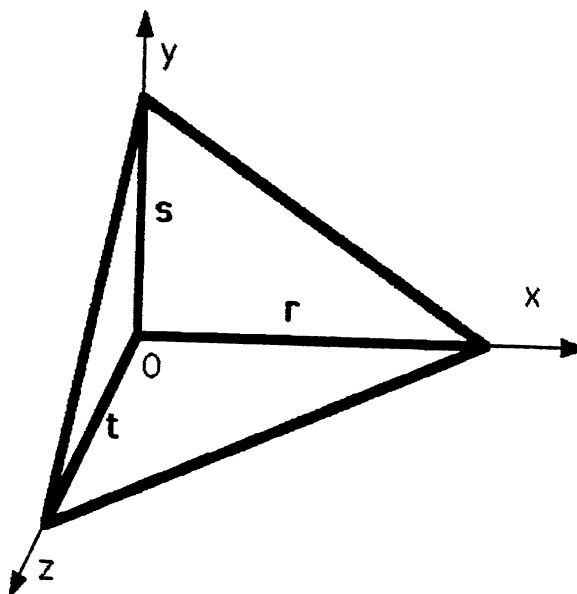


Figure 5.3: Calculating a Volume Integral: A Special Case

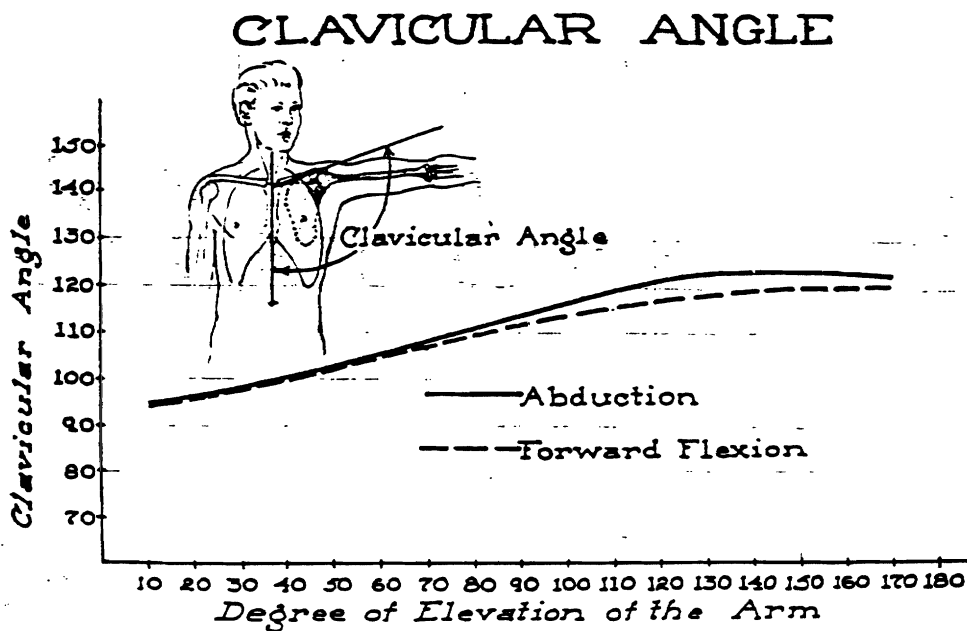


Figure 5.4: Inman's graph showing relation between clavicle and humerus elevations

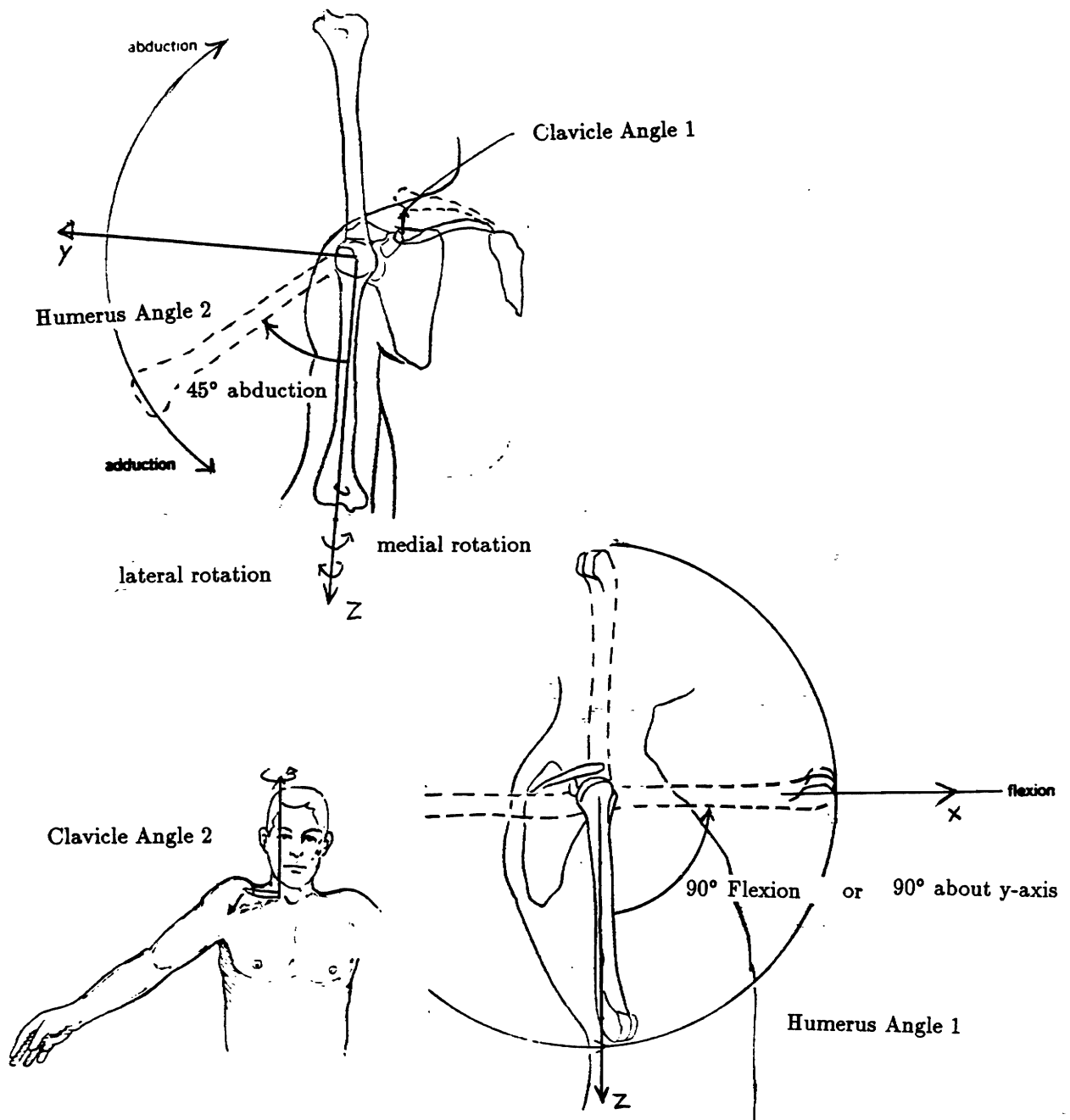


Figure 5.5: Definitions of Joint Angles in the Shoulders

# Chapter 6

## Conclusion

The objective of this project was to improve realism and facilitate interaction in computer modeled human motion. As the goal has two parts, so does the project. The first part, improving the facility for dynamic simulation, and developing a model to be used with it was only partly successful. Mover, the dynamics interface for the Jack environment, is functional but not convenient. Human motion depends heavily on the active, conscious movements of the individual. Such movements are especially hard to develop with a preprocessor. Too much time is spent waiting for results that could often be better spent designing a motion keyframe by keyframe.

Ideally, dynamic simulation should be applied interactively. A situation would be modeled and as the simulation progressed, the user could change the parameters of the simulation and so could have some better measure of control of the resulting motion. This is quite different from the crash-test family of simulators that includes Mover.

In defense of the Mover system, it is well suited for generating motions that are strongly governed by external forces and moments. And, if it is some day coupled with a strength model, its range of applications could expand beyond being mostly a crash simulator to include active tasks requiring strength and planning.

The development of the body model to accompany Mover was fairly successful. The biomechanics literature contained a large enough body of studies that I feel confident the models for joint stiffness which I present are adequate for most simple body models. The only major weakness of the joint model is mentioned in chapter 4. Muscles that span more than one joint have a profound impact on limb flexibility

and active strength. Any future models should accommodate this.

The prototype figure suggested at the end of chapter 4 clearly suffers from the lack of a sufficient body of data. Many values in the prototype were merely estimates and at best, the prototype is a “Frankenstein’s monster” with data for different parameters taken from different experimental subjects (some of whom were cadavers). As I stressed in chapter 4, the prototype is only meant to serve as reasonable template, and in that context, it is successful.

The other part of this project, improving the interactive tools was somewhat more successful. The interactive center of mass routine and coupled shoulder positioning algorithm both suggest a trend towards increasing the level of realism in the development of interactive positioning tools. I am most encouraged by the linked motion algorithm which suggests a vast number applications that could link on variable in the graphics environment to another. Figures could flip light switches and influence the lighting model. Body segments could change shape as a function of joint angle thus mimicking the flexing of muscles. Many such applications could grow out of this simple idea.

The automatic mass and inertia generator is surprisingly robust. With the single restriction on the ordering of polygon vertices and the assumption of constant density, the inertia of *any* object that can be represented with a psurf can be calculated. Objects don’t even have to be simply connected.

Future projects that can build on this work are:

- Modify the Mover system to handle trajectory motion problems.
- Add collision detection and modeling as either an interactive tool, or a part of the dynamics preprocessor.
- Expand on the idea of adding realism by linking the behavior of some things to the behavior of others.
- Collect a coherent set of joint stiffness data from a significant population and run a verification study on the joint models.

# Bibliography

- [1] Schaffa, R.B., "Dynamic Analysis of Spatial Mechanisms", Ph.D. Dissertation, Department of Mechanical Engineering and Applied Mechanics, The University of Pennsylvania (1984).
- [2] Paul, B., "Program DYSPAM" in **Multibody Systems Handbook** Springer Verlag (to be published).
- [3] Dvir, Z. and Berme, N., "The Shoulder Complex in Elevation of the Arm: A Mechanism Approach", *Journal of Biomechanics*, Vol. 11, 1978, pp. 219-225.
- [4] King, A.I. and Chou, C.C., "Mathematical Modelling, Simulation and Experimental Testing of Biomechanical System Crash Response," *J. of Biomechanics*, Vol. 9, 1976, pp. 301-317.
- [5] Prasad, P., "An Overview of Major Occupant Simulation Models," SAE Paper (?) No. 840855.
- [6] Robbins, D.H., "Three-dimensional Simulation of Advanced Automotive Restraint Systems," Paper No. 700421, 1970 Int. Automotive Safety Conf. Compendium. P-30, SAE.
- [7] Robbins, D.H., Bennett, Jr., R.O. and Bowman, B.M., "User-oriented Mathematical Crash Victim Simulator," Proc. 16th Stapp Car Crash Conference, pp 128-148, SAE, 1972.
- [8] Young, R.D., "A Three-dimensional Mathematical Model of an Automobile Passenger," Texas Transportation Inst. Res. Report, pp. 140-142, 1970.

- [9] Furusho, H. and Yokoya, K., "Analysis of Occupant's Movements in Head-on Collision," *Trans. Soc. Automotive Engr. Japan*, Vol. 1, 1970, pp. 145-155.
- [10] Huston, R.L., Hessel, R. and Passerello, C., "A Three-dimensional Vehicle-man Model of Collision and High Acceleration Studies," Paper No. 740275. Society of Automotive Engineers Inc. 1974.
- [11] Bartz, J.A., "A Three-dimensional Computer Simulation of a Motor Vehicle Crash Victim, Phase 1 - Development of the Computer Program," Calspan report No. VJ-2978-V-1, July 1971.
- [12] Bartz, J. A. and Butler, F. E., "A Three-dimensional Computer Simulation of a Motor Vehicle Crash Victim, Phase 2 - Validation Study of the Model," Calspan Report No. VJ-2978-V-2, December 1972.
- [13] Fleck, J.T., Butler, F.E. and Vogel, S.L., "An Improved Three-dimensional Computer Simulation of Motor Vehicle Crash Victims," Volumes I-IV, Report Nos. DOT-HS-801507 thru -510, 1974.
- [14] Fleck, J. T. and Butler, F.E., "Development of an Improved Computer Model of the Human Body and Extremity Dynamics," Report No. AMRL-TR-75-14, July 1975.
- [15] Wismans, J., Maltha, J., Melvin, J. W. and Stalnaker, R. L., "Child Restraint Evaluation by Experimental and Mathematical Simulation," SAE 791017, Proc. 23rd. Stapp Car Crash Conference, SAE, Warrendale, Pennsylvania, 1979.
- [16] Wismans, J., Maltha, J., Van Wijk, J. J. and Janssen, E. G., "MADYMO - A Crash Victim Simulation Computer Program for Biomechanical Research and Optimization of Designs for Impact Injury Prevention," AGARD-meeting, Cologne, Germany, April, 1982.
- [17] King, A. I., "A Review of Biomechanical Models," *J. of Biomechanical Engineering*, Vol. 106, 1984, pp.97-104.
- [18] Onyshko, S., and Winter, D.A., "A Mathematical Model for the Dynamics of Human Locomotion," *J. of Biomechanics*, Vol. 13, 1980, pp. 361-368.

- [19] Engin, A. E., "Kinematic and Passive Resistive Properties of the Human Elbow Complex," *J. of Biomechanical Engineering*, Vol.109, 1987, pp. 318-323.
- [20] Inman, V. T., Saunders, J. B. and Abbott, L. C., "Observations on the Function of the Shoulder Joint," *J. Bone Jt. Surg.*, Vol. 26a, 1944, pp. 1-30.
- [21] Engin, A. E., "On the Biomechanics of the Shoulder Complex," *J. Biomechanics*, Vol. 13, 1980, pp. 575-590.
- [22] Engin, A. E., "On the Damping Properties of the Shoulder Complex," *J. Biomechanical Engineering*, Vol. 106, 1984, pp. 360-363.
- [23] Yoon, Y. S., Mansour, J. M., "The Passive Elastic Moment at the Hip," *J. Biomechanics* Vol. 15, 1982, pp. 905-910.
- [24] Blankevoort, L., Huiskes, R. and De Lange, A., "The Envelope of Passive Knee Joint Motion," *J. Biomechanics*, Vol. 9, 1988, pp. 705-720.
- [25] Kinzel, G.L. and Gukowski, L.J., "Joint Models, Degrees of Freedom and Anatomical Motion Measurement," *J. of Biomechanical Engineering*, Vol. 105, 1983, pp. 55-62.
- [26] Grood, E.S. and Suntay, W.J., "A Joint Coordinate System for the Clinical Description of Three Dimensional Motions: Application to the Knee," *J. Biomechanical Engineering*, Vol. 105, 1983, pp. 136-144.
- [27] Dooley, M., "Anthropometric Modeling Programs - A Survey," *IEEE Computer Graphics and Applications*, Vol. 2, no. 9, 1982, pp. 17-25.
- [28] Wilmert, K.D., "Visualizing Human Body Motion Simulations," *IEEE Computer Graphics and Applications*, Vol. 2, no. 9, 1982, pp. 35-38.
- [29] Wilhelms, J., "Using Dynamic Analysis for Realistic Animation of Articulated Bodies," *IEEE Computer Graphics and Applications*, Vol. 7, no. 6, 1987, pp. 12-27.
- [30] Girard, M., "Interactive Design of 3D Computer-animated Legged Animal Motion," *IEEE Computer Graphics and Applications*, Vol. 7, no. 6, 1987, pp. 39-51.

- [31] Grosso, M.R., Quach, R.D. and Badler, N. I., "Anthropometry for Computer Animated Human Figures," Technical Report, Department of Computer and Information Science, The University of Pennsylvania (1989).
- [32] Phillips, C.B., "Programming with Jack", unpublished manual, University of Pennsylvania Dept. of Computer and Information Sciences, 1988.
- [33] Phillips, C.B., and Badler, N.I., "Jack: A Toolkit for Manipulating Articulated Figures", Proceedings of the ACM SIGGRAPH Symposium on User Interface Software, Banff, Alberta, Canada, 1988.
- [34] Strassberg, E.R., "User Interface and Animation Processor for Dynamics of Spatial Mechanisms Simulated via DYSPAM", M.S.E Thesis, University of Pennsylvania Department of Mechanical Engineering and Applied Mechanics, 1989.
- [35] Goldstein, H., Chapter 4 and Appendix B of **Classical Mechanics, Second Edition**, Addison-Wesley, 1980.
- [36] Budiansky, B. , Chapter 4 of **Handbook of Applied Mathematics** Pearson, C.E. ed. VanNostrand Rienhold Company, 1974.
- [37] Badler, N.I., O'Rourke, J., and Kaufman, B., "Special Problems in Human Movement Simulation," *ACM Computer Graphics*, 1980.



# Appendix A

## Figure Definition for the Prototype Body

```
figure {
  attribute attribute3 {
    rgb = (1.00,0.37,0.00);
  }
  attribute attribute11 {
    rgb = (1.00,0.37,0.00);
  }
  attribute attribute13 {
    rgb = (1.00,0.37,0.00);
  }
  attribute attribute15 {
    rgb = (1.00,0.37,0.00);
  }
  segment right_lower_leg {
    psurf = "ecalf.pss";
    attribute = attribute3;
    mass = 4000.77g;
    inertia = (1199091.8750,1453679.8750,292090.1563);
    site proximal->location = trans(0.00cm,0.00cm,0.00cm);
    site distal->location = trans(0.00cm,0.00cm,36.80cm);
    site CM->location = trans(2.61cm,0.01cm,23.73cm);
  }
  segment left_lower_leg {
    psurf = "ecalf.pss";
    attribute = attribute3;
    mass = 4000.77g;
    inertia = (1199091.8750,1453679.8750,292090.1563);
    site proximal->location = trans(0.00cm,0.00cm,0.00cm);
    site distal->location = trans(0.00cm,0.00cm,36.80cm);
    site CM->location = trans(2.61cm,0.01cm,23.73cm);
  }
  segment right_upper_leg {
    psurf = "eupleg.pss" ;
    attribute = attribute3;
    mass = 8205.55g;
  }
}
```

```

    inertia = (992266.1875,1013080.3125,156199.0156);
    site proximal->location = trans(0.00cm,0.00cm,0.00cm);
    site distal ->location = trans(0.00cm,0.00cm,43.40cm);
  }
  site CM->location = trans(0.00cm,0.00cm,18.07cm);
}
segment left_upper_leg {
  psurf = "eupleg.pss" ;
  attribute = attribute3;
  mass = 8205.55g;
  inertia = (992266.1875,1013080.3125,156199.0156);
  site proximal->location = trans(0.00cm,0.00cm,0.00cm);
  site distal->location = trans(0.00cm,0.00cm,43.40cm);
  site CM->location = trans(0.00cm,0.00cm,18.07cm);
}
segment right_lower_arm {
  psurf = "elowarm.pss";
  attribute = (attribute3,attribute11);
  mass = 1815.81g;
  inertia = (365474.6875,361975.3750,13135.1484);
  site proximal->location = trans(0.00cm,0.00cm,0.00cm);
  site distal->location = trans(0.00cm,0.00cm,28.80cm);
  site CM->location = trans(-0.01cm,0.00cm,17.51cm);
}
segment left_lower_arm {
  psurf = "elowarm.pss";
  attribute = (attribute3,attribute13);
  mass = 1815.81g;
  inertia = (365474.6875,361975.3750,13135.1484);
  site proximal->location = trans(0.00cm,0.00cm,0.00cm);
  site distal->location = trans(0.00cm,0.00cm,28.80cm);
  site CM->location = trans(-0.01cm,0.00cm,17.51cm);
}
segment right_upper_arm {
  psurf = "euparm.pss";
  attribute = attribute3;
  mass = 2297.40g;
  inertia = (165567.2500,171408.9688,19422.7441);
  site proximal->location = trans(0.00cm,0.00cm,0.00cm);
  site distal->location = trans(0.00cm,0.00cm,33.40cm);
  site CM->location = trans(-0.03cm,0.02cm,13.99cm);
}
segment left_upper_arm {
  psurf = "euparm.pss";
  attribute = attribute3;
  mass = 2297.40g;
  inertia = (165567.2500,171408.9688,19422.7441);
  site proximal->location = trans(0.00cm,0.00cm,0.00cm);
  site distal->location = trans(0.00cm,0.00cm,33.40cm);
  site CM->location = trans(-0.03cm,0.02cm,13.99cm);
}
segment lower_torso {

```

```

psurf = "eltorso.pss";
attribute = (attribute3,attribute13);
mass = 3469.32g;
inertia = (232921.1406,124958.4453,303026.2500);
site proximal->location = trans(0.00cm,0.00cm,0.00cm);
site distal->location = trans(0.00cm,0.00cm,13.10cm);
site rlateral->location = xyz(-180.00deg,0.00deg,0.00deg)
* trans(0.00cm,-6.12cm,0.00cm);
site llateral->location = xyz(-180.00deg,0.00deg,0.00deg)
* trans(0.00cm,6.12cm,0.00cm);
site CM->location = trans(0.00cm,0.00cm,6.17cm);
}
segment center_torso {
psurf = "echest.pss";
attribute = (attribute3,attribute15);
mass = 13169.60g;
inertia = (9540146.0000,9199660.0000,1203154.8750);
site proximal->location = trans(0.00cm,0.00cm,0.00cm);
site distal->location = trans(0.00cm,0.00cm,47.60cm);
site utproximal->location = trans(0.00cm,0.00cm,47.60cm);
site utdistal->location = trans(0.00cm,0.00cm,47.60cm);
site utleft->location = xyz(-90.00deg,0.00deg,0.00deg)
* trans(0.00cm,0.00cm,47.60cm);
site utright->location = xyz(90.00deg,0.00deg,0.00deg)
* trans(0.00cm,0.00cm,47.60cm);
site rstdistal->location = xyz(90.00deg,0.00deg,0.00deg)
* trans(0.00cm,0.00cm,50.00cm);
site lstdistal->location = xyz(-90.00deg,0.00deg,0.00deg)
* trans(0.00cm,0.00cm,50.00cm);
site rcl_lateral->location = xyz(-180.00deg,0.00deg,0.00deg)
* trans(0.00cm,-17.20cm,50.00cm);
site lcl_lateral->location = xyz(-180.00deg,0.00deg,0.00deg)
* trans(0.00cm,17.20cm,50.00cm);
site CM->location = trans(0.09cm,-0.08cm,33.76cm);
}
segment body_root {
mass = -1.00g;
site distal->location = xyz(-90.00deg,0.00deg,-90.00deg)
* trans(0.00cm,0.00cm,0.00cm);
site left->location = xyz(-90.00deg,0.00deg,0.00deg)
* trans(0.00cm,0.00cm,0.00cm);
site right->location = xyz(90.00deg,0.00deg,0.00deg)
* trans(0.00cm,0.00cm,0.00cm);
site floor->location = xyz(45.00deg,90.00deg,45.00deg)
* trans(0.00cm,0.00cm,-94.10cm);
site base->location = trans(0.00cm,-94.10cm,0.00cm);
}
joint waist {
connect lower_torso.distal to center_torso.proximal;
type = R(0.00,0.00,1.00)*R(1.00,0.00,0.00)
*R(0.00,1.00,0.00);
stiff = (1.00,1.00,1.00);
}

```

```

}
joint root_torso {
    connect body_root.distal to lower_torso.proximal;
    type = R(1.00,0.00,0.00)*R(0.00,1.00,0.00)
        *R(0.00,0.00,1.00);
}
joint left_shoulder {
    connect center_torso.lcl_lateral to left_upper_arm.proximal;
    type = R(0.00,0.00,1.00)*R(1.00,0.00,0.00)
        *R(0.00,1.00,0.00);
    stiff = (1.00,1.00,1.00);
}
joint right_shoulder {
    connect center_torso.rcl_lateral to right_upper_arm.proximal;
    type = R(0.00,0.00,-1.00)*R(-1.00,0.00,0.00)
        *R(0.00,1.00,0.00);
    stiff = (1.00,1.00,1.00);
}
joint right_elbow {
    connect right_upper_arm.distal to right_lower_arm.proximal;
    type = R(0.00,1.00,0.00);
    stiff = (1.00);
}
joint left_elbow {
    connect left_upper_arm.distal to left_lower_arm.proximal;
    type = R(0.00,1.00,0.00);
    stiff = (1.00);
}
joint right_hip_joint {
    connect lower_torso.rlateral to right_upper_leg.proximal;
    type = R(0.00,0.00,-1.00)*R(-1.00,0.00,0.00)
        *R(0.00,1.00,0.00);
    stiff = (1.00,1.00,1.00);
}
joint left_hip_joint {
    connect lower_torso.llateral to left_upper_leg.proximal;
    type = R(0.00,0.00,1.00)*R(1.00,0.00,0.00)
        *R(0.00,1.00,0.00);
    stiff = (1.00,1.00,1.00);
}
joint right_knee {
    connect right_upper_leg.distal to right_lower_leg.proximal;
    type = R(0.00,-1.00,0.00);
    stiff = (1.00);
}
joint left_knee {
    connect left_upper_leg.distal to left_lower_leg.proximal;
    type = R(0.00,-1.00,0.00);
    stiff = (1.00);
}
root = body_root.base;
}

```

## “Optional” DYSPAM input file for Prototype Figure

```
1 1
-4.27e+06 1.96e+08 6.76e+06 -8.6e+08
800000 0.0
0.0 0.0 0.0
1 2
4.96e+08 -1.53e+09 -1.2e+09 2.41e+09
800000 0.0
0.0 0.0 0.0
1 3
-4.65e+07 1.36e+09 2.89e+07 -9.26e+08
800000 0.0
0.0 0.0 0.0
3 1
9163170 86218689 -141554609.8 52665465.7
800000 0.0
0.0 0.0 0.0
3 2
-22066180 305488491.3 -508802156 299064608
800000 0.0
0.0 0.0 0.0
3 3
16496420 -84232817 -185215934 -94045474
800000 0.0
0.0 0.0 0.0
4 1
-9163170 -86218689 141554609.8 -52665465.7
800000 0.0
0.0 0.0 0.0
4 2
-22066180 305488491.3 -508802156 299064608
800000 0.0
0.0 0.0 0.0
4 3
-16496420 84232817 185215934 94045474
800000 0.0
0.0 0.0 0.0
5 3
10140000 292240000 -61388500 2144230000
800000 0.0
0.0 0.0 1.7
```

6 3  
10140000 292240000 -61388500 2144230000  
800000 0.0  
0.0 0.0 1.7  
7 1  
1.31e+06 -1.36e+09 -8.42e+07 2.18e+09  
800000 0.0  
0.0 0.0 0.0  
7 2  
331806720 -1446000000 1882000000 -827600000  
800000 0.0  
0.0 0.0 0.0  
7 3  
-2.91e+07 -3.83e+08 1.34e+08 1.06e+09  
800000 0.0  
0.0 0.0 0.0  
8 1  
-1.31e+06 1.36e+09 8.42e+07 -2.18e+09  
800000 0.0  
0.0 0.0 0.0  
8 2  
331806720 -1446000000 1882000000 -827600000  
800000 0.0  
0.0 0.0 0.0  
8 3  
2.91e+07 3.83e+08 -1.34e+08 -1.06e+09  
800000 0.0  
0.0 0.0 0.0  
9 3  
-155000000 845000000 -1240000000 492000000  
800000 0.0  
0.0 0.0 1.553  
10 3  
-155000000 845000000 -1240000000 492000000  
800000 0.0  
0.0 0.0 1.553