

ANIMATION OF HUMAN MOTION:
AN INTERACTIVE TOOL

A Thesis

Submitted to The Department of Computer
Engineering and Information Science
and The Institute of Engineering and Science
of Bilkent University
in Partial Fulfillment of The Requirements
For The Degree of
Master of Science

By

Syed Kamran Mahmud

January, 1991

Thesis
TR
897.7
.M34
1991

ANIMATION OF HUMAN MOTION: AN INTERACTIVE TOOL

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER
ENGINEERING AND INFORMATION SCIENCE
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By

Syed Kamran Mahmud

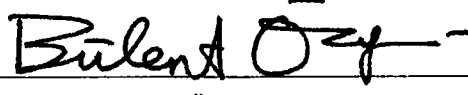
January, 1991

Syed Kamran Mahmud
tarafidan taqdim qayim qayim,

TR
897.7
-M34
1991

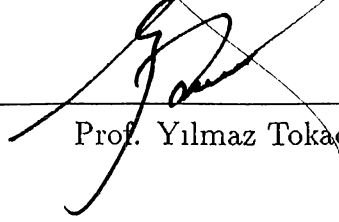
B 7967

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.



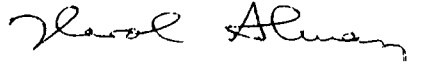
Prof. Bülent Özgüç (Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.



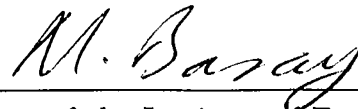
Prof. Yılmaz Tokad

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.



Assoc. Prof. Varol Akman

Approved by the Institute of Engineering and Science:



Prof. Mehmet Baray, Director of the Institute of Engineering and Science

ABSTRACT

ANIMATION OF HUMAN MOTION: AN INTERACTIVE TOOL

Syed Kamran Mahmud

M.S. in Computer Engineering and Information Science

Supervisor: Prof. Bülent Özgüç

January, 1991

The goal of this work is the implementation of an interactive, general purpose, human motion animation tool. The tool uses *parametric key-frame animation* as the animation technique. Different abstractions of motion specification in *key-frame* generation are explored, and a new notion of *semi goal-directed animation* for generating key-frame orientations of human body is introduced to resolve the tradeoff between animator and machine burden in choosing a level of motion specification.

Keywords: Human Animation, Classical Animation, Computer Animation, Simulation, Motion Specification

ÖZET

İNSAN HAREKETİNİN CANLANDIRILMASI: ETKİLEŞİMLİ BİR ARAÇ

Syed Kamran Mahmud

Bilgisayar Mühendisliği ve Enformatik Bilimi Bölümü Yüksek
Lisans

Tez Yöneticisi: Prof. Bülent Özgüç

Ocak, 1991

Bu çalışmada amaç, genel maksatlı, etkileşimli bir insan hareketin modelleme aracı geliştirmektir. Araç canlandırılma tekniği olarak parametrik anahtar çerçeve tekniğini kullanmaktadır. Anahtar çerçeveleri oluşturmanın değişik soyutlamaları araştırılmakta ve insan vücudunun anahtar çerçeve noktalarını bulmada yeni bir yaklaşım olan “yarı amaç-yönlendirmeli canlandırılma” tanıtılmaktadır. Bu yeni yöntemle, hareket tanımlamının derecelendirmesinin yapılmasında animatörle makine arasındaki yük dağılımı sorunu çözümlenebilecektir.

Anahtar Kelimeler: İnsan Canlandırılma, Geleneksel Canlandırılma, Bilgisayarlı Canlandırılma, Benzetişim, Hareket Tanımlama

ACKNOWLEDGEMENT

I would like to thank my supervisor Professor Bülent Özgüç for his guidance and encouragement during the development of this thesis.

I am grateful to Professor Yılmaz Tokad for helping me in the proof of the appendix B, and for his remarks and comments on the thesis.

I am also grateful to Associate Professor Varol Akman for providing me with important references and for his remarks and comments on the thesis.

I express my gratitude to Associate Professor Kemal Oflazer who provided me with references that were quite helpful.

My sincere thanks are due to my parents, sisters, and brothers for their moral support.

Finally, I appreciate my friends Veysi İşler, Ahmet Arslan, M. S. Ali, Nihal Mutluay, and all others who helped and cooperated during my thesis.

TABLE OF CONTENTS

| | | |
|----------|--|-----------|
| 1 | INTRODUCTION | 1 |
| 2 | HUMAN BODY MODEL | 4 |
| 2.1 | HUMAN BODY MODELING TECHNIQUES | 4 |
| 2.2 | SEGMENT MODEL | 6 |
| 2.3 | JOINT MODEL | 8 |
| 2.3.1 | Joint Motions | 8 |
| 2.3.2 | Joint Limits | 12 |
| 3 | HUMAN BODY ANIMATION | 16 |
| 3.1 | ANIMATION TECHNIQUES | 16 |
| 3.1.1 | Goal-Directed Animation | 18 |
| 3.1.2 | Algorithmic Animation | 20 |
| 3.1.3 | Procedural Animation | 21 |
| 3.1.4 | Key-Frame Animation | 21 |
| 3.2 | PARAMETRIC KEY-FRAME ANIMATION | 23 |
| 3.2.1 | Motion Specifications | 24 |

| | | |
|----------|--|-----------|
| 3.2.2 | Inbetweening | 52 |
| 3.2.3 | Integrating Algorithmic Animation for Realistic Motion . | 56 |
| 4 | AOHM IMPLEMENTATION | 60 |
| 4.1 | DATA STRUCTURES | 61 |
| 4.1.1 | Joint | 61 |
| 4.1.2 | Frame | 63 |
| 4.1.3 | Film | 63 |
| 4.1.4 | Menu Command | 64 |
| 4.1.5 | Coordinate Axes | 66 |
| 4.2 | USER INTERFACE | 67 |
| 5 | FUTURE DIRECTIONS | 71 |
| 6 | CONCLUSIONS | 73 |
| | APPENDICES | 79 |
| A | ARC-CIRCLE INTERSECTIONS | 80 |
| B | AOHM USER'S MANUAL | 86 |

LIST OF FIGURES

| | | |
|-----|--|----|
| 2.1 | Three types of human body models | 5 |
| 2.2 | Human body model in AOHM | 6 |
| 2.3 | Body segment model | 7 |
| 2.4 | Body joints | 9 |
| 2.5 | Computational model of a joint | 10 |
| 2.6 | Table of human body joint motions | 12 |
| 2.7 | Joint limit boundary | 14 |
| 2.8 | Joint limit configurations in space | 15 |
| 3.1 | Human body orientation by joint modification | 27 |
| 3.2 | State changes in joints | 28 |
| 3.3 | Joint positioning | 30 |
| 3.4 | Arm model for positioning wrist | 33 |
| 3.5 | Arm triangle | 35 |
| 3.6 | Elbow circle in wrist positioning | 36 |
| 3.7 | Elbow circle and the joint limit boundary | 38 |
| 3.8 | Different cases of elbow circle and joint limit boundary | 39 |

| | | |
|------|--|----|
| 3.9 | Transformed elbow circle and the joint limit boundary | 40 |
| 3.10 | Different cases of the intersections between an arc and a circle, both on the same sphere | 42 |
| 3.11 | Intersection of two circles lying on a sphere | 44 |
| 3.12 | Body orientation by menu-command | 50 |
| 3.13 | Modified menu-command effect | 50 |
| 3.14 | Inbetweening | 55 |
| 3.15 | Actual and interpolated ANKLE paths | 57 |
| 3.16 | Improved interpolated ANKLE path | 58 |
| 3.17 | Algorithmic ANKLE path | 58 |
| 4.1 | Main window of the AOHM tool | 68 |
| 4.2 | Multi-view environment of the AOHM tool | 69 |
| A.1 | Arc-circle intersections | 81 |
| B.1 | Main window of AOHM | 87 |
| B.2 | Button panel in the main mode of AOHM | 88 |
| B.3 | Multi-view environment of AOHM | 89 |
| B.4 | Scale pop-up window | 89 |
| B.5 | Button panel in the frame mode | 90 |
| B.6 | Rotation pop-up window | 91 |
| B.7 | Translation pop-up window | 91 |
| B.8 | Pop-up window for menu-command mode | 91 |

| | |
|---|----|
| B.9 Menu-command parameter modification | 92 |
| B.10 Define-Command pop-up window | 93 |
| B.11 Cursor positions in the multi-view environment | 94 |
| B.12 Joint positioning mode window | 95 |
| B.13 Joint parameter information mode | 96 |
| B.14 Information about the joint parameters | 96 |
| B.15 Button panel in film mode of AOHM | 97 |

1. INTRODUCTION

Animating articulated bodies like humans and animals has always been a problem for the computer. There are a couple of reasons:

First, humans and animals cannot easily be modeled by mathematical and geometric techniques used in computer graphics because of their peculiar shape. Thus, very realistic models are hardly possible. Work has been carried out for close to realistic models, for instance Fetter's work on the visual effects of hemispheric projections mentioned in [38] and models based on data obtained by anthropometrists, Dooley discussed in [38] and [19], but such models are very expensive from the point of view of rendering and CPU time, thus cannot easily be used interactively. Very realistic models have been tried for parts of the body instead of the whole body, for example a hand has been implemented for animating the task of grasping some object [18]. Such difficulties dictate the use of a less realistic model while working interactively. One solution is to use interactive skeleton technique [8] (i.e., use skeleton model while interacting and later build the complex model over it).

Second, specification and control of motion in human figure animation have always been a challenge. It is very difficult for the animator to generate key-frames by orienting the model as the limb coordination is complex and he has to control and specify each joint angle. Degree of realism lies in the hands of the animator. Motion specification problem is explored in different aspects by many researchers [1, 3, 4, 7, 15, 16, 23, 43, 44].

One suggested solution minimizes the animator's load in motion specification, viz. the notion of *goal-directed animation*. In this technique the animator generates the whole motion sequence by simple English commands. Here the

system does all the motion control and planning. The mechanism of motion control and planning has to be embedded in the system at least once. For example in [7] the animator generates the motion of human walk simply by the command WALK. This is a very good approach from the animator's viewpoint, but a little thought reveals two critical problems. First, since the motion is controlled and planned by the computer, mechanisms or algorithms are to be found for all types of motion included in the system. This takes us to another area of research, namely *Robotics*. Second, no matter how good and efficient techniques we find, it will hardly be possible for us to incorporate the wide span of motions depicted by human being.

The focus of our research is on this issue. As a solution to the problem of level of motion specification, we have suggested a new abstraction level called *semi goal-directed animation*, in which the animator generates the key-frames of a motion sequence by selecting from the pool of system and user-defined commands. This level of motion specification simplifies animator's job to a great extent. It does not add much burden to the system or machine for planning and controlling the human body motions. It optimizes the trade-off between the man and the machine load in selecting a level of motion specification.

A rather old but considerable amount of information on human modeling and animation can be found in [10]. We have many references from the proceedings of "*Graphics Interface '86 Vision Interface '86*", because we think that this single issue is quite comprehensive in the field of our research.

After the above discussion, it is worth mentioning where we need such a general purpose tool. The main application would be in the areas of ergonomics, choreography and robotics. It can be useful for ergonomic applications if complex reach algorithms are incorporated. Choreographers working with dance and skating will also find this useful. It may be helpful to computer movie-makers. Finally, movements of a human like robot can be studied with this tool.

In this thesis we start in Chapter 2 with the issue of modelling human body in such a way that it is suitable for the animation purpose.

In Chapter 3, we start with the definition of animation and animation techniques. The main technique concerning us, *parametric key-frame animation*, is discussed in depth, with the main focus on different abstractions of motion specification. We have suggested a taxonomy for motion specification level in key-frame animation to understand and explore the issue more conveniently.

Chapter 4 discusses the implementation of AOHM (Animation of Human Motion) focusing on the main data structures.

Chapter 5 includes the future directions followed by our conclusions.

Appendix B is the user manual of AOHM.

2. HUMAN BODY MODEL

For modeling any object the details have to be well-understood, especially if the object is complex, viz. human body. We have, therefore, tried to explore the model of human body together with some information of the nature of the human body itself.

Modeling human body realistically is one of the most challenging problems. There are two main reasons for this: (i) geometrical and mathematical models used in computer graphics are not very suitable for the shape of the human body, (ii) the movement of joints is difficult to model, particularly because of the peculiar muscle actions.

Ideally, realistic rendering can only be achieved by *3-D rotoscopy*, digitizing by hand the joint coordinates of all body segments from at least two orthogonal views recorded on film or video. This approach is tedious. It is important in biomechanics research and work is being done for its automation. Another problem with *rotoscopy* is that only those motions can be animated which are performed at least once by somebody.

2.1 HUMAN BODY MODELING TECHNIQUES

There are three general methods for modeling the human body in 3-D [38] :

Stick Figures: These are like a skeleton and consist of a collection of line segments attached by joints representing human body joints. Stick figures are unrealistic (figure 2.1(a)).

Surface Figures: These are basically stick figures surrounded by surfaces that consist of planar or curved patches (figure 2.1(b)).

Volume Figures : Volume figures are the body models in which the body is decomposed into several primitive volumes. Usually cylinders, ellipsoids, and spheres are used as primitive volumes (figure 2.1(c)).

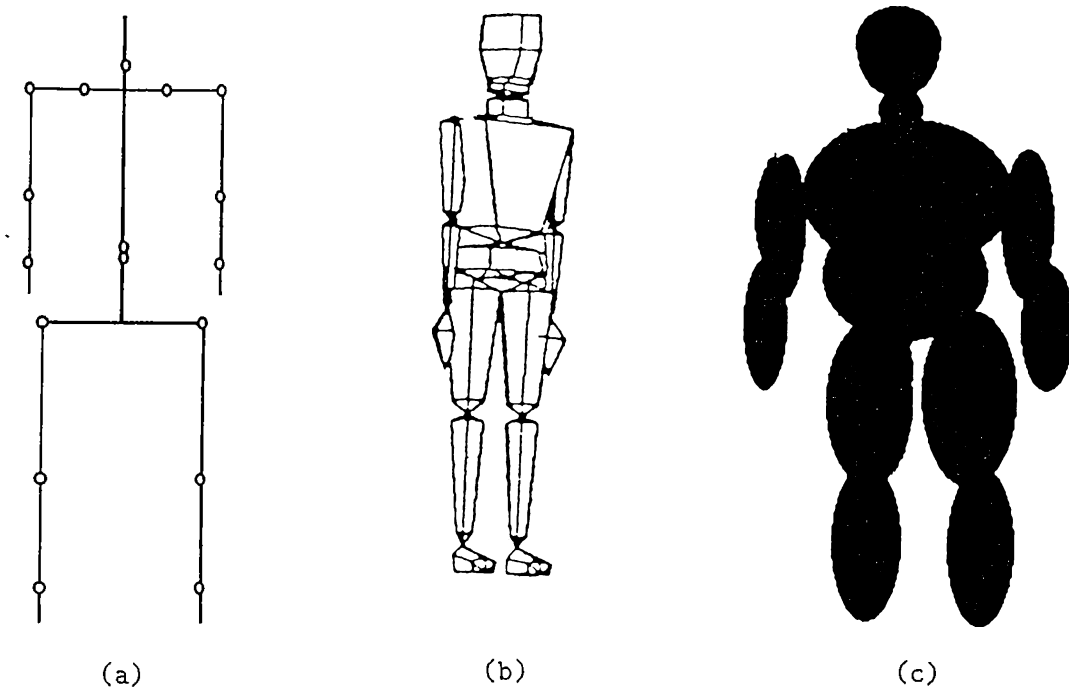


Figure 2.1: (a) Stick model (b) Surface model (c) Volume model

Practically, the level of detail included in a human body model is dictated by its purpose. The model that we are using is a kind of wire framed surface model, but without any planar or curved patch (figure 2.2). Actually each body segment is represented by a set of equidistant rectangles, one on top of the other, thus giving a surface feature. The choice for such a model is made because it requires less displaying time compared to most other surface models.

There is always a trade off between realism and the time consumed in

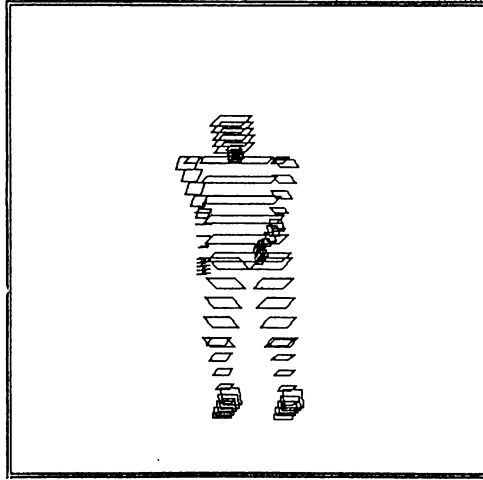


Figure 2.2: Human body model in AOHM

displaying. Stick model consumes the minimum displaying time but realism is almost totally absent because depths are difficult to evaluate and several movements like twists are impossible to represent. Thus our choice is almost a minimally acceptable one in terms of realism and almost optimal in terms of displaying time.

2.2 SEGMENT MODEL

We have assumed the body segments to be rigid. Our body segment model is in fact like a rectangular box, consisting of five equidistant rectangles one on the top of the other. For each modification of the body segment one has to transform 20 (i.e. 5×4) vertices (figure 2.3(a)). Each transformation requires six multiplications and six additions (three for each x and y). What we are doing is that we transform 8 (i.e. 2×4) vertices (figure 2.3(b)), the vertices of the top and the bottom rectangles, and while displaying we compute the vertices of the intermediate rectangles by:

$$x = x1 + u(x2 - x1)$$

$$y = y1 + u(y2 - y1)$$

where 'u' is a parameter taking values 0.25, 0.50, and 0.75, and $(x1, y1)$ and $(x2, y2)$ are the corresponding vertices of the bottom and the top rectangles.

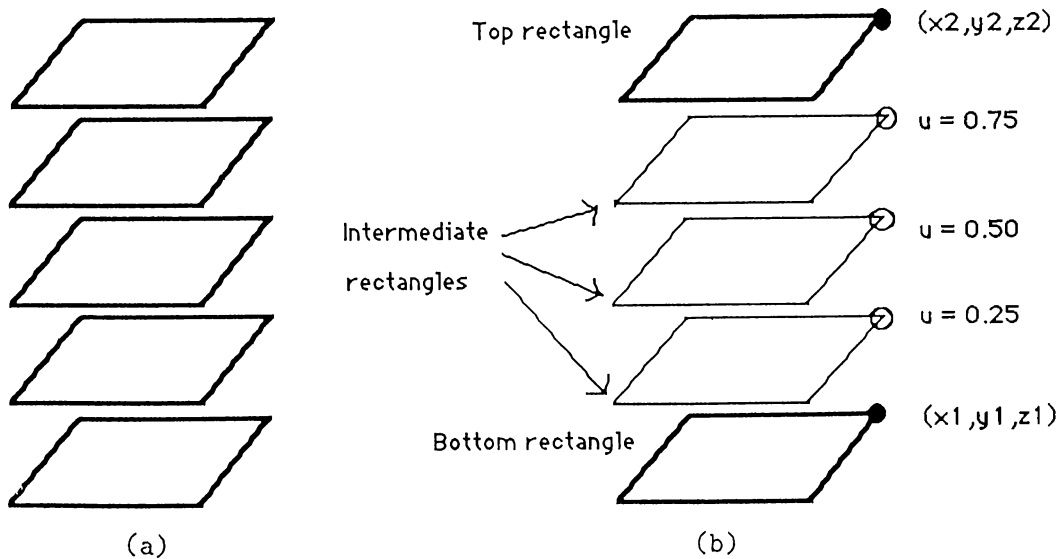


Figure 2.3: (a) Model of the body using five rectangles, i.e. 20 vertices are transformed. (b) A body segment model in which only 8 vertices are transformed. ● : Actual vertices that are transformed and ◯ : Interpolated.

This parametric computation requires only two floating point additions and one floating point multiplication. Thus we save more than 500 floating point multiplications and more than 250 floating point additions in each modification of the orientation of the body.

In fact the choice of a model is application or purpose dependent. Our choice is reasonable for the purpose of interactive simulation tool. The body model typically consists of a tree structure, either body joints as nodes connected by body segments or body segments as nodes connected by body joints. Both schemes have been tried, examples of the first scheme are CAR model and the original bubbleman model [23]. The second scheme is used by Boeman (Boeing Corporation) [23], Bubbleman (University of Pennsylvania) [23], CAPE (Pacific Missile Test Center) [23], and others.

In the tree structure there is always a joint (or segment) taken as the root. In our model WAIST (upper torso) is selected as root joint (segment). This choice is natural as most of the limbs meet here.

2.3 JOINT MODEL

Modeling a joint means finding a computational model for representing the different types of motions depicted by them. The first step in modeling a joint is to decide on the depth of details to be included in the model, i.e., the number of joints the model will have. The more the number of joints the better the model is approximated. For all practical purposes, one cannot model to the depth of all the joints in the human body. In our model we have approximated with 18 joints (figure 2.4).

Human joints in general are basically complex and idiosyncratic, as the limits of the degrees of freedom of their depicted motions are not constant. The limit of a joint parameter representing one degree of freedom may be a non-linear function of the instantaneous values of the joint parameters representing other degrees of freedom of the motions of that joint. This interrelated nature of joint limits causes a real challenge while they are modeled and leads to a trade-off between accuracy and uniformity [23].

The basic function of a joint is to connect two body segments (or links). Joints act as a pivot for segments to achieve different orientations while moving. The segment movement is restricted about a joint. During a single joint movement, one (proximal) segment is stationary and the other (distal) segment moves with respect to the *proximal segment*. Proximal means closer to the root (which only acts as stationary for all joints belonging to human body) in the hierarchy. In fact a segment can function as both proximal and distal if it belongs to two or more joints.

Every joint is assigned a unique identifier for referring them during specification of motions or body orientations. The joint identifiers used in AOHM are ordinary names and not anatomical ones.

2.3.1 Joint Motions

In modeling joint motions it is better to reduce the freedom of motions embedded in each joint to one or more independent parameters [23]. This eliminates

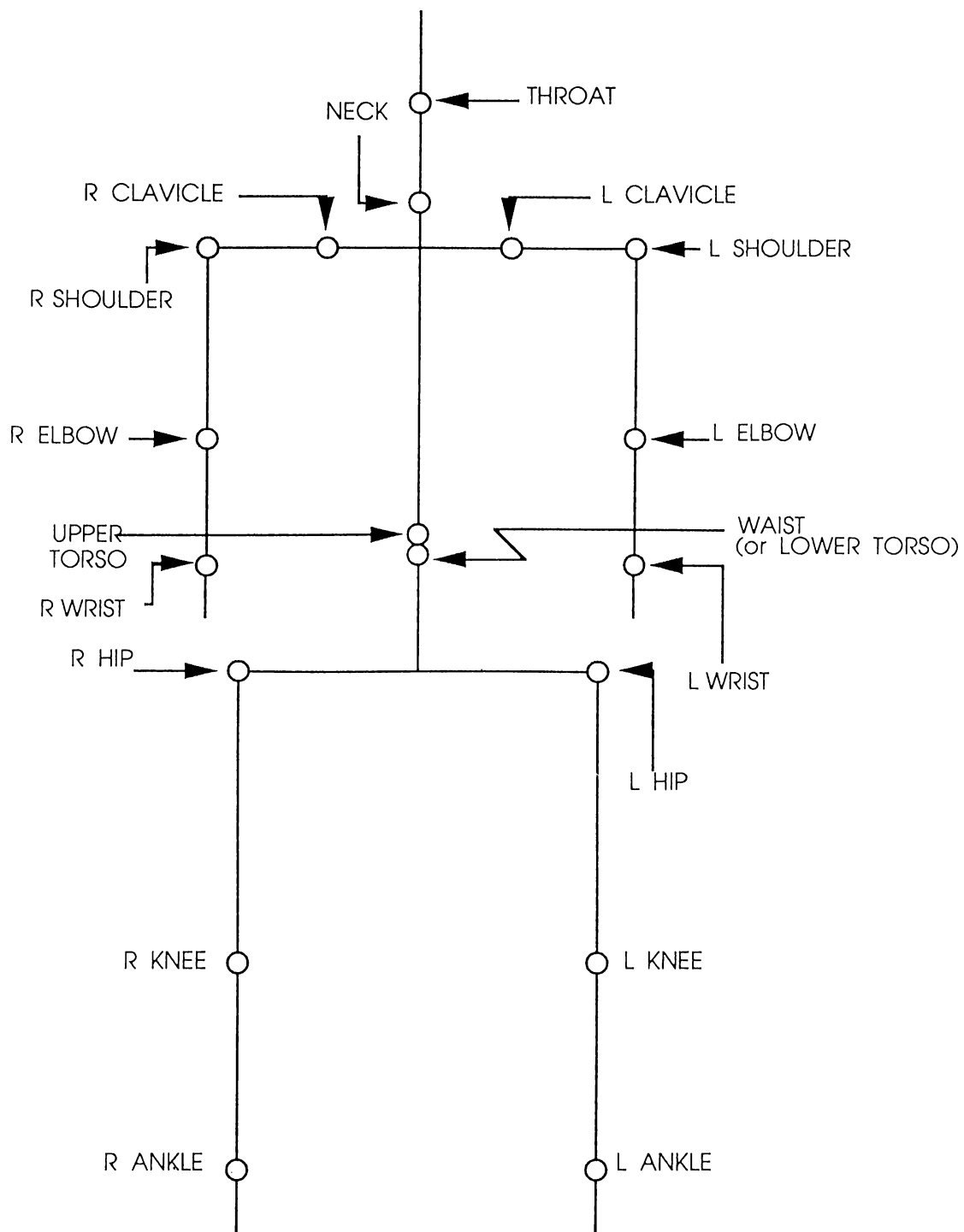


Figure 2.4: Joints included in the model as a tree structure

inconsistencies, especially while interpolating or generating inbetweens from the key-frames. To achieve this, a small number of motion types can be defined and we can let the joints have one or more of these motions.

There are different types of joint motions with variable degrees of freedom [19]. A joint has up to three degrees of freedom [9]. Three degrees of freedom can be constructed and motions of lesser degrees of freedom can be achieved by restricting this model to a subset of degrees of freedom allowed. Thus, simple joints such as fingers (hinge joints), and complex joints, such as shoulders (ball-and-socket joints) can be simulated. A joint can move independently of all other joints except its parent and grandparents in the tree structure; this independent movement of the joints actually defines the joint's actual trajectory.

Usual anatomical concepts about joint motions cause ambiguity in modeling. Thus the mechanical approach of revolute and spherical joints plays an significant role in describing joint motions of the human body.

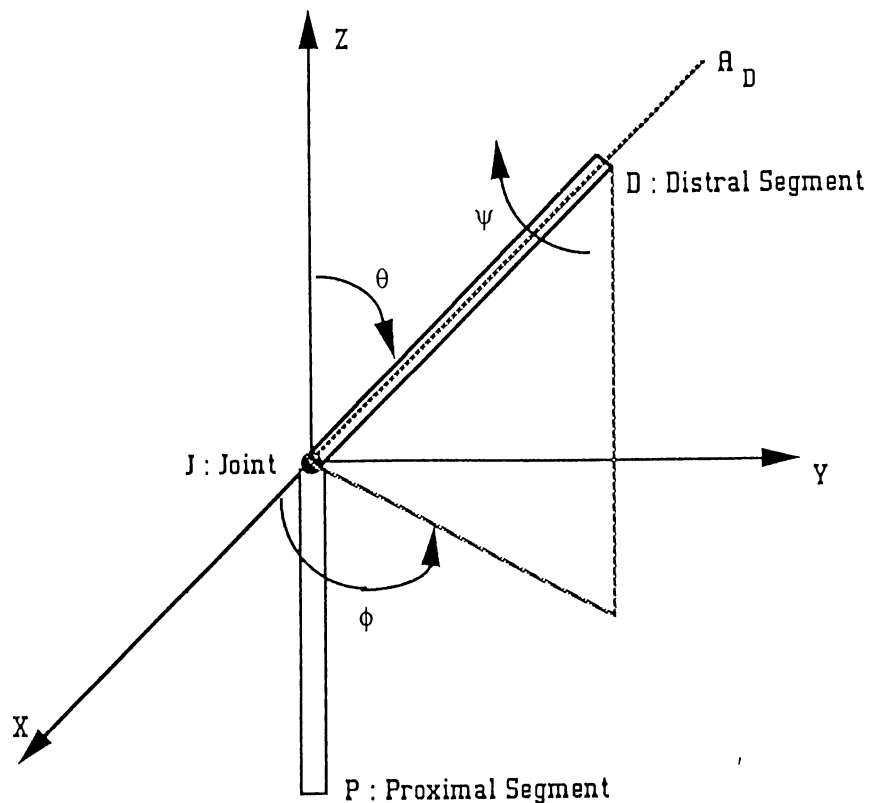


Figure 2.5: Computational model of a joint

A spherical joint has three degrees of freedom, and the three parameters (or variables) used can be defined by numerous means and using a different coordinate system (figure 2.5).

The proximal segment **P** is fixed in the coordinate system (spherical in AOHM) originating at the joint **J**, and axis A_D of distal segment **D** is a line through the origin. The position (or orientation) of the distal segment is given by ϕ , θ , and ψ . ϕ and θ give the description of the direction of the axis A_D , and ψ describes the amount of rotation of distal segment **D** about A_D . The type of motion achieved by the variation of first two parameters (ϕ , θ) is called *spherical motion*, as it specifies the position of the sibling joint J_S of the joint **J** at the other end of distal segment **D** in spherical coordinate system with a fixed radius or r of spherical coordinates (r , θ , ϕ), and the motion by (ψ) is called *twist* [23]. The motion depicted by shoulder having three degrees of freedom, needs to be assigned both of these motion models. For instance, positioning the elbow (J_S) by modifying the state of shoulder **J** joint is an example of spherical motion. ψ corresponds to the twisting of the upper arm about its axis, *twist motion*.

Besides spherical and twist motions, there is another important and common motion type known as *flexion motion*. Examples of *flexion* motions are seen in the knee and elbow flexion. The distinction between *flexion* and *twist* is useful for clarity of visualization [23]. *Flexion* motions can be modeled via the spherical motion model by restricting it to only one degree of freedom (θ) as flexion motion has only one degree of freedom. Thus we have the following three joint motion models:

- Spherical
- Twist
- Flexion

To understand the nature of motions depicted by the joints given in figure 2.4 or about the joints included in the human body model AOHM implementation, table in figure 2.6 is provided:

| Joints | Types of Motion |
|------------|------------------|
| ankle | spherical |
| clavicle | spherical |
| elbow | flexion, twist |
| hip | spherical, twist |
| knee | flexion, twist |
| neck | flexion, twist |
| shoulder | spherical, twist |
| throat | spherical |
| uppertorso | spherical, twist |
| waist | spherical, twist |
| wrist | spherical |

Figure 2.6: Table of human body joint motions

In AOHM only *spherical* and *flexion* are included; we shall incorporate *twist* in the near future.

2.3.2 Joint Limits

We need to consider joint limits in order to make the motion of the human body more realistic. If it had not been there, the animator would have oriented the human body in any way he would have preferred, treating the human body like any linked body without any restrictions on joint, and many orientations would have appeared that are not possible.

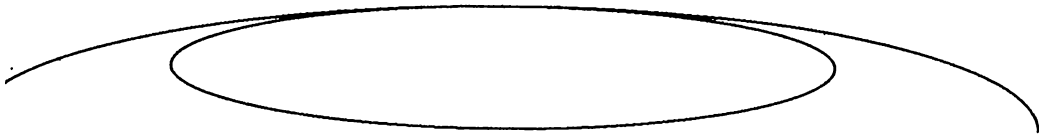
Joint limits to be discussed in this section is based on the joint motion models mentioned in the previous section, namely, *flexion*, *twist*, and *spherical*.

Flexion and twist motions have a single degree of freedom, so joint limits are just constants (a minimum value and a maximum value). These constant limit values are based on the assumption that the extreme values (limits) do not depend on the current value of any other joint variable; this is not true

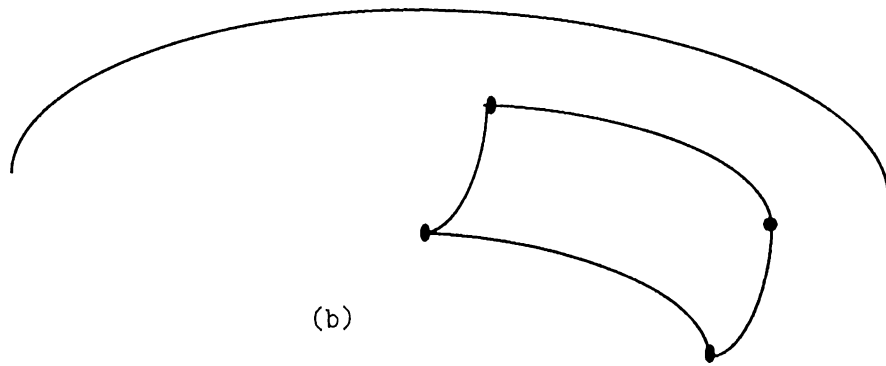
in general. But for many practical purposes, this assumption does not cause much trouble.

Joint limits for spherical motion is not a trivial one. Actually the distal end of the distal segment is forced to lie on a sphere about a joint. On imposing joint limits the distal end is restricted to lie within some patch on the sphere. The boundary of this patch is called the *joint limit curve*. When the two variables are treated independently, curves of figures 2.7(a) and (b) are observed. In figure 2.7(a) the first parameter (θ) is restricted to certain values (maximum and minimum) and the second parameter (ϕ) is allowed to span through the whole of its domain. In figure 2.7(b) both of the parameters have certain extreme values (maximum and minimum). These types of limit approaches are not close approximations to the actual limit boundaries of human joint limit curves but they, being very cheap to implement, are widely used and we have also implemented this scheme in AOHM. Figure 2.6(c) shows another approach which is a closer approximation to the actual case. In this scheme the joint limit boundary is approximated by a spherical polygon, consisting of points on the sphere connected by arcs.

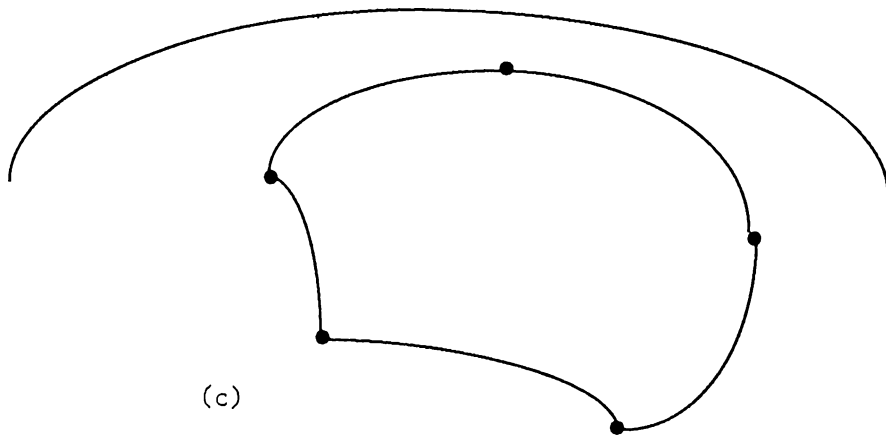
For a joint depicting both spherical motion and twist motion (a shoulder is an example of such a joint), different joint limit boundary approximation approaches can best be understood by describing the set of allowable values for the joint variables in the 3-D configuration space defined by the variables ϕ , θ , and ψ . In most general case where all the joint limits are dependent on the current values of other variables (parameter), the configuration is an arbitrary blob in space (figure 2.8(a)). In figure 2.8(b), the most simple case is shown when all the three joint limits are constant values, a maximum and a minimum. In figure 2.8(c) third parameter (ψ), i.e., twist motion has constant value limits, but the limits of spherical motion are not constant but interrelated.



(a)



(b)



(c)

Figure 2.7: (a) Independent joint limits, (b) Another independent joint limits, (c) Joint limit curve approximated by a polygon on the surface of the sphere.

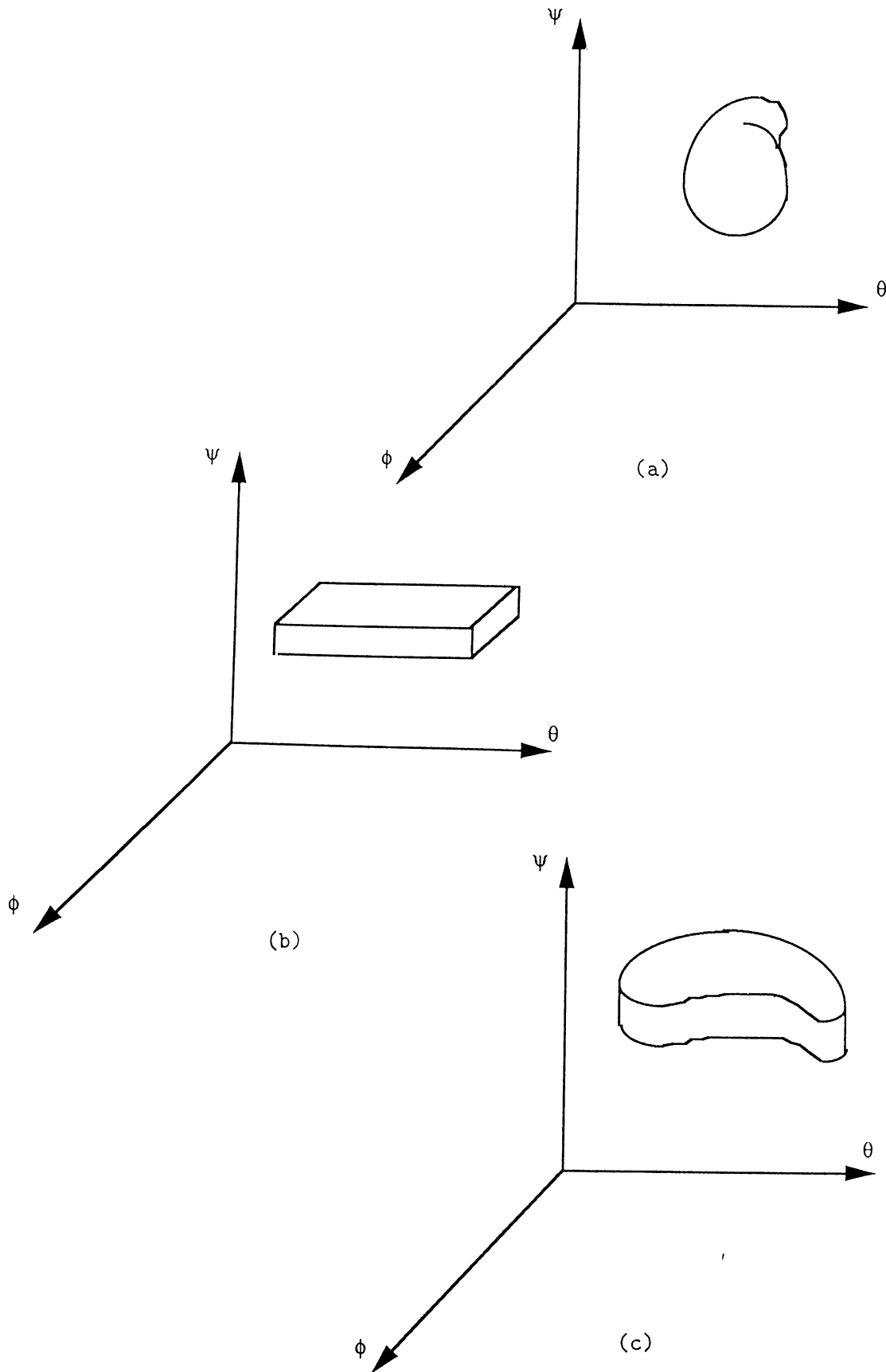


Figure 2.8: (a) All joint limits are coupled, (b) All joint limits are independent, (c) ϕ and θ limits are coupled but ψ limit is independent.

3. HUMAN BODY ANIMATION

“Above all, animation is the art of movement. The accomplished animator can bring life to just about anything - a series of drawings or a tin can” [24].

The etymological definition of ‘Animation’ can be stated as ‘giving life to something’; that something can be a still picture drawn manually on a piece of paper or by computer on a display device. So for our purpose ‘Animation’ is making the pictures of human body drawn on a still frame appear to move, making use of the *persistence of vision* phenomenon of the human eye.

When computer animation is discussed a few questions come to mind: What will be the animation technique at display level? How one is going to show the still pictures? What will be the animation technique at the level of creation of *sequence frame* (sequence frame is the set of still pictures to be displayed while animation)? How each of the frame of the sequence is going to be generated or to put in practical terms? How the motion is going to be specified, so that the burden could be brought to an acceptable ratio?

3.1 ANIMATION TECHNIQUES

There is a variety of techniques for animation, among these we shall be discussing those which can be considered for the animation of articulated bodies like human body. As mentioned above, animation techniques can be and should be explored both at the display level and at the level of frame sequence creation.

There are basically two animation techniques as far as the display of a 3-D picture is concerned:

- Frame by frame
- Draw and display

In the first technique the whole frame sequence is prepared and put into the memory and then each frame is displayed one by one. This technique requires large memory for even short animations. A partial solution is to not save every frame, but rather save the differences between the consecutive frames. This slows down the rate of display.

In the second technique, each frame of the sequence is prepared when it is to be displayed. Hardware plays an important role because the drawing mechanism needs to be really fast, to achieve smooth and realistic motion. An important issue to be noted here is that when the frames are to be drawn, and how much information is ready in the created frame. This preprocessing can be anything like calculating the color values, or transforming some data points, or even finding a suitable transformation for the frame. The more manipulation a frame requires before it is ready to be drawn, the slower the display speed will be. Fast animations can be achieved by completely preprocessing the frame information beforehand. For complex figures like human body, to do all the manipulation beforehand is not quite feasible from the memory point of view, especially if the sequence is permitted to be modifiable.

As far as the generation of frame sequence is concerned, we propose the following classification of animation techniques, especially for human figure or any articulated body in general:

- Goal-directed animation
- Algorithmic animation
- Procedural animation
- Key-frame animation

3.1.1 Goal-Directed Animation

All frames of the motion sequence for a particular motion is generated by the machine in response to user's simple English-like commands, such as 'walk', 'run', 'jump', etc. [11]. The motion planning and control is done by the machine and the animator does nothing except specifying a command. We know that say, human walk, is not a unique motion; rather there are hundreds of different kinds of walks depending on the step size, speed, personal dimensions, etc., so just giving a command like 'walk' we cannot achieve different types of walks. As a solution the concept of *parametric goal-directed* animation is used. Here, together with the command 'walk' some parameters like step-size, speed etc. [7] are allowed.

In goal-directed animation, be it parametric or nonparametric, the animator gets rid of the planning and control of human motion and motion in general. In this technique the main problem is how the machine is going to plan and control the motion.

Procedural methods can be used which has a knowledge base for motion [34], but this is not very realistic for most cases. This is because the quality of motion depends on the amount of information embedded in the knowledge base. This issue becomes more critical when one is trying to develop a general-purpose tool for human motions, as a huge amount of information is needed. This technique can be implemented as a language base [11], or with parametric functions.

Kinematic control is another method and is the one mostly used in designing control mechanisms for human motion planning. Although this also is not realistic, it still gives acceptable performance. The basic tools of a kinematic approach is position, displacement, and velocity of the object. Sometimes acceleration is also used. In kinematic approach forces and torques are not considered.

Dynamic control is the one that gives the most realistic results but computationally it is expensive. In dynamic control, besides position, displacement, and velocity, energy, force, and torque which tend to cause the motions are also

considered and the *center of mass* plays an important role in motion planning. But calculating the center of mass of human figures is not trivial. The main reason for dynamic approach being computationally so expensive is the typical structure of human body, the articulated nature of body segments. So mass analysis is needed to be done for each body segment individually; same is true for force and torque calculations. Apart from being expensive, there is another problem with this technique: the animator has to supply the information about the forces and torques.

To get a flavour of the dynamic approach let us consider the following:

An angle x of a joint is to be modified to y . One can calculate that torque T applied to a joint by [1]:

$$T(x) = \alpha e^{(\beta \operatorname{sgn}(x-y)(x-y)-1)}$$

The parameters α and β determine the strength of the torque applied at the joint and they can be controlled by the animator in parametric goal-directed animation. In general, one can divide a motion sequence in phases either for the whole body or for groups of body segments and for each phase one can derive a system of non-linear equations by using the Lagrange method. The Lagrange equations for a system with n degrees of freedom can be written as:

$$\frac{d}{dt} \left(\frac{\delta L}{\delta \dot{q}_r} \right) - \frac{\delta L}{\delta q_r} = F_{q_r}$$

$$\begin{aligned} \text{where } r &= 1, \dots, n \\ L &= \text{Lagrangian} = T - V \\ T &= \text{Kinetic energy} \\ V &= \text{Potential energy} \\ q_r &= \text{Generalized coordinate} \\ F_{q_r} &= \text{Generalized force} \end{aligned}$$

This is used in [7]; what is done is that the walk motion of the legs are divided into two phases, viz. *stance* and *swing*. *Stance* means that the leg is touching the ground and *swing* means that the leg is in motion, that is, off the ground. The Lagrangian method is applied to the individual phases of the walk motion, namely stance and swing. The phase analysis is first done independently at the

level of body parts. At the end of this analysis the dependencies are propagated to other related body parts in terms of forces and torques.

Practically the dynamical control is never used as the sole control mechanism, rather kinematic control is used and decorated with dynamics whenever needed [7, 8, 15, 16, 33, 43, 44].

3.1.2 Algorithmic Animation

Motion is described algorithmically. Actually physical laws are applied to parameters of the human body, e.g., joint angles. Control of these laws may be provided to the system by procedural methods as in MIRA [14] or by interactively as in MIRANIM [14]. In fact one can specify almost any law for the parameters after building a reasonable input mechanism.

These laws may be based on kinematic approach, i.e., positions, displacement, and velocity, or dynamic approach, incorporating force and torque in addition to the kinematic criteria. As discussed before, dynamic approach is expensive so here also mostly the laws based on kinematics are applied and dynamics is incorporated whenever needed. These laws can also be specified as functions defining the trajectories of a joint in a motion sequence. But for these a sound understanding of the trajectories of the human body joints during a particular motion is necessary [3, 27, 28, 29].

Whatever law is used, *algorithmic animation* is expensive in general because a computation of the algorithmic function is done for each parameter of every joint while generating a single frame and same is true for each frame of a whole motion sequence.

Algorithmic animation, being computationally expensive, is hardly ever used. Instead, in most applications it is integrated with other techniques which are comparatively cheaper. This is discussed in section 3.2.3.

3.1.3 Procedural Animation

This technique includes describing animation by an animation language (e.g., CINEMIRA-2 [13]) or by scripts (in script systems). Procedural animation's main basis is the knowledge base of motions, and the language, functions, and scripts all consult and depend on it. There can be different abstractions for the level of description of motion in procedural animation; the highest level is the same as goal directed animation.

Procedural animation, being heavily dependent on the knowledge base, cannot be used in tools designed for the wide range of human motions, as it is difficult to include all the details of the whole span of human motion.

3.1.4 Key-Frame Animation

Key-frame animation is one of the oldest animation techniques still in use because of its inherent nature that suits interactive environment. In this technique the animator typically orients and positions the human body interactively, designating a sequence of configurations, called the *key-frame* and in most of the cases also describing the time instances of their occurrences. Then the system automatically generates the intermediate frames, known as *inbetweens*, based on the sequence of key-frames supplied by the animator.

Interpolating between related key images allows the animation of change of shape and distortion in general, and change of orientation and position in the case of human motions. Actually it permits a direct method for specifying the motion or action, in contrast to the mathematically defined distortion that requires trial and error. A desired result is achieved by iterated experiments. One quality observed in key-frame animation is that it is in a way analogous to conventional animation, making it easier for a classically trained animator.

Image-Based Inbetweening

The inbetweens are obtained by interpolating the key-frame images. This technique is called *image based key-frame animation* by Steketee and Badler [35] or *shape interpolation* by Zeltzer [13]. This is an old technique introduced by Burtnyk and Wein [13].

Parametric Inbetweening

This is a better way to compute the inbetweens. Instead of the images, the joint parameters of the human body are interpolated. This is called *parametric key-frame animation*. It is also known as *key-transformation animation*. In a parameter model, the animator creates key-frames by specifying the appropriate set of parameter values, and then the parameters are interpolated and images for the inbetweens are obtained from the interpolated parameters. Parametric key-frame animation is actually the technique used in AOHM.

There are a few issues needed to be resolved while implementing this technique. First, how will the animator specify the motions? How will the animator produce key-frames? Will he (and not the machine) modify the joint parameters to yield an orientation of human body in a key-frame? Another way can be that the animator will just give a command and the system will do all the job and prepare all the key-frames for the animation, as in goal-directed animation.

Second, we have the problem of interpolating the parameters of the body specified in the key-frame for yielding inbetweens. So one needs to explore some methods while designing a tool based on key-frame animation.

Third, the animator has almost no control over the inbetweens, the interpolation of the key-frames. Quite often unrealistic motion results in the inbetweens if special care is not taken. Thus this issue cannot be neglected if one is aiming at designing a tool which should not produce very unrealistic motions.

3.2 PARAMETRIC KEY-FRAME ANIMATION

This is the technique used in the implementation of our AOHM. In parametric key-frame animation, the animator interactively orients and positions the human body for the generation of each frame, called *key-frames*, i.e., some important orientations and positions of the human body at selected intervals during the time span of a desired motion. Orienting and positioning the human body means specifying the value for body parameters like a particular degree of freedom variable of some joint, directly or indirectly. Directly means modifying each joint parameter explicitly for generating a key-frame of a motion sequence. This is a laborious task for the animator and the machine almost has nothing to do, this is the lowest level of motion specification. Indirectly means some higher level of motion specification that is finally translated into the lowest level by the machine, thus parameters are modified implicitly.

Once the animator completes the creation of key-frames for the motion sequence, the system automatically, based on some interpolating method generates the rest of the frames, called *inbetweens*, i.e., the frames lying between the consecutive key-frames.

There is a problem with this technique: The animator does not have total control over the inbetweens generated by the system. It is almost impossible for an interpolating method to be intelligent enough to produce the inbetweens realistically for a wide range of motions such as the span of human motions. Thus there is a need of some mechanism to upgrade the quality of the inbetweens.

Parametric key-frame is chosen for AOHM because it is inherently very suitable for interactive motion specification which is one of the main themes of AOHM. Intelligent motion controlling and planning algorithm is also not really needed in this technique.

3.2.1 Motion Specifications

Motion control is a central problem in computer animation [43]. The specification and control of motion in human figure animation has always been a challenge [7], and many different approaches are being made to resolve it. Badler [3] approaches the issue by exploring the criteria to a good or acceptable movement representation. Badler [5] also suggests natural language as an artificial intelligence application to motion specification. English-commands [11] and parameterized goals [7] suggest the motion specification in the highest level. There are other approaches like using functions [43] and body orientations and positioning [9] as lower level specifications.

After analyzing all the above researches and suggestions for motion specification levels one thing is worth noting; as the motion specification level gets higher, that is to say as the animator's job is simplified, the computer does more work and the motion control and planning algorithm gets more intelligent and complex. Moreover, the animator's control over motion generation becomes less. Thus if in a particular motion that is supported by the system in a high level of motion specification the animator desires to have some minor adjustments, then it is almost impossible, especially if in the parameterized goal approach the available parameters cannot support that modification. Moreover in high level of motion specifications wide range of motions cannot be supported. It is also worth noting that if the animator is given all the control i.e. while generating a key-frame play with any parameter (degree of freedom variable of joint motion) of the human body and no higher level motion specification is supported, then even for the generation of a single motion frame the animator would have to do very laborious work and the machine would almost be idle, but here almost any motion is possible. Thus there is a trade-off between the level of control and the level of ease the animator should have in motion specification.

Considering the above trade-off a need for a tool that supports different abstractions of motion specification is felt and for the implementation of such a tool there is a need for a taxonomy of motion specification levels. We propose the following taxonomy:

- Joint parameter modification
- Joint positioning
- Semi goal directed approach
- Goal directed approach

As we go down the classification the level increases, the animator's job gets simpler, and the computer's work gets more complicated.

Joint Parameter Modification

This is the lowest level of motion specification. Here the computer does not have to do any translation of the animator's command, it is in the most primitive form. But for specifying motion, rather orientation and position the animator's job is burdensome.

Translation: This primitive specification is stated as a triplet:

$$T = (t_x, t_y, t_z)$$

where t_x, t_y , and t_z are displacements, Δx , Δy , and Δz in coordinates \mathbf{x} , \mathbf{y} , and \mathbf{z} of the human body (as a whole articulated object) defined in Cartesian coordinate system, respectively. It should be noted here that translation cannot be specified for a joint or segment as they make up the articulated figure.

Rotation: This is the second type of primitive specification of motion and is the most used one. It can be specified as

$$R = (\text{joint-name}, \text{angle}, \text{axis})$$

where 'joint-name' is the joint identifier whose state is desired to be modified, 'angle' gives the amount of change in the value of a joint parameter desired, and 'axis' actually determines which degree of freedom should be modified. In AOHM *axis* takes only two values; 1 to specify θ and 2 to specify ϕ . A state is

a set of values of the parameters (θ, ϕ) of joint motion. Now to understand the state and how the state is modified or in other words how primitive specification 'rotation' is made, consider figure 3.1((a) and (b)). The joint parameters are actually spherical coordinate ordinates.

In figure 3.1(a) the joint 'RSHOULDER' (right shoulder) has a state $S_1(\theta = 40^\circ, \phi = 65^\circ)$ and rotation

$$R = (RSHOULDER, -20^\circ, 1)$$

changes the state from S_1 to $S_2(\theta = 20^\circ, \phi = 65^\circ)$. Actually the primitive specification says that change the state of joint 'RSHOULDER', decrementing the value of its first degree of freedom by 20° .

The coordinates of joint or location of joint in space given by triplet (x, y, z) is not included in the state definition. This can be better understood if figure 3.2((a) and (b)) is studied.

In figure 3.2(a) there are three joints A, B and C with states (θ_a, ϕ_a) , (θ_b, ϕ_b) , and (θ_c, ϕ_c) , and joint positions (x_a, y_a, z_a) , (x_b, y_b, z_b) , and (x_c, y_c, z_c) respectively. The joints A, B, and C constitute a linked configuration, being a part of linked figure like human body. The joint B is connected to the joint A through the body segment AB. A similar relation exists between the joints B and C. Now modifying the state of the joint A to (θ'_a, ϕ'_a) , the coordinates of the joint B will get modified to (x'_b, y'_b, z'_b) and that of C to (x'_c, y'_c, z'_c) . Here it is worth noting that the relative orientation of the body segment BC with respect to the body segment AB is unchanged, thus the state of the joint B is unchanged. Similarly the state of the joint C is unchanged. This isolation of position (x, y, z) from the state (θ, ϕ) is made because when the position changes with respect to its siblings, it only means that one of its parent or grand parent joint has changed its state or in other words some comparatively proximal joint has modified its orientation. Thus any change in the state of a joint is propagated through its siblings to the end-effectors by position change.

In AOHM joint limits are also implemented but in their simplest form and we hope to make it closer to realistic joint limit curves in the future. Joint limits for each of the joint parameters (degree of freedom) is a pair of maximum

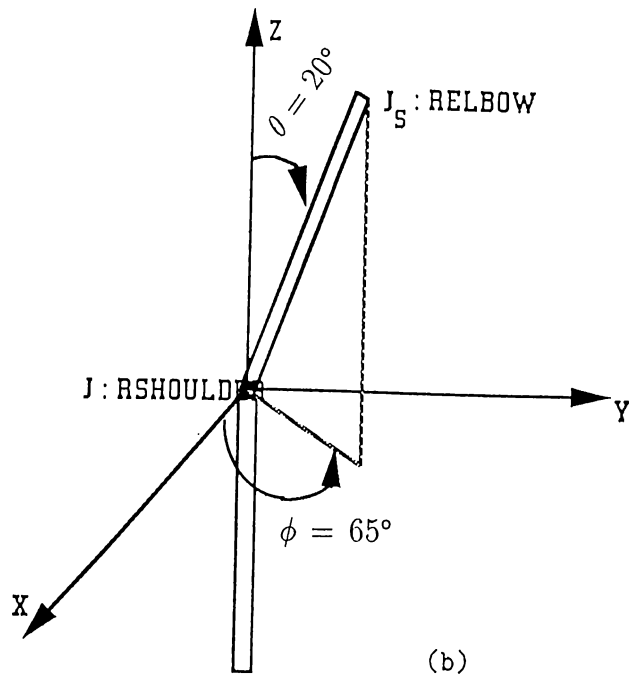
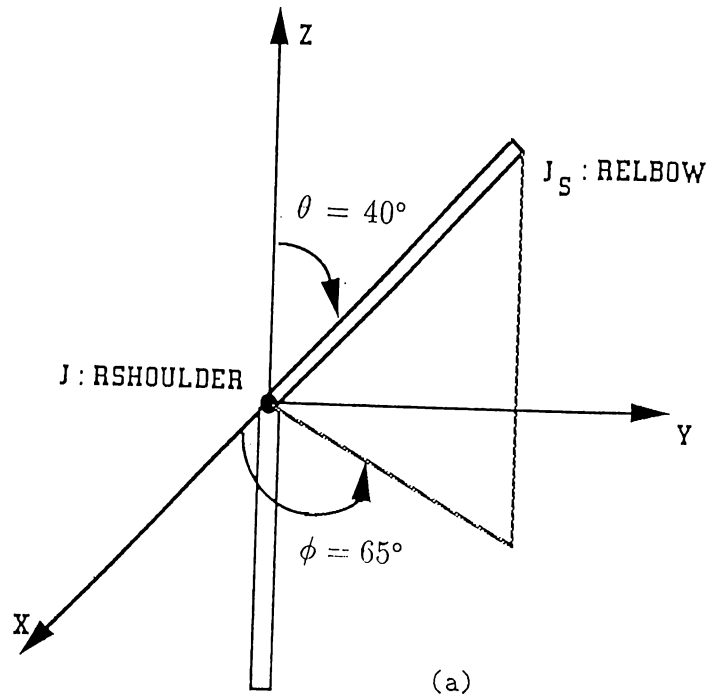


Figure 3.1: (a) Initial state of the joint with $\theta = 40^\circ$ and $\phi = 65^\circ$ (b) Final state of the joint after the rotation $R = (RSHOULDER, -20^\circ, 1)$ i.e. $\theta = 20^\circ$ and $\phi = 65^\circ$.

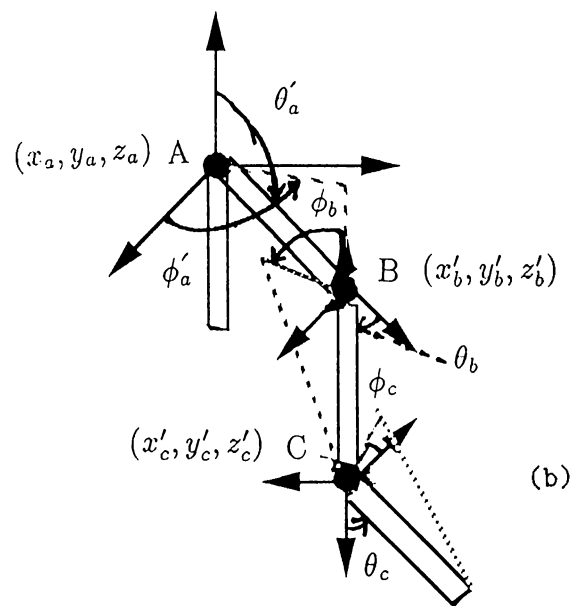
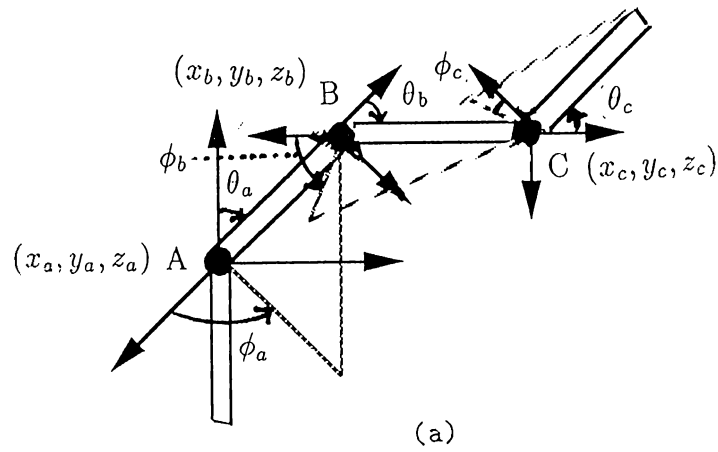


Figure 3.2: (a) Initial states and coordinates of the joints A, B, and C (b) Final state of the joint A and different coordinate values of the joints B and C.

and minimum. Whenever a primitive motion specification is made the system checks whether the new value is within the limits. If it is, then the value of the joint parameter is modified. Otherwise, a message is given to the animator that the limit is violated and the modification is not allowed.

There is a problem of induced twist in the body segments or joints; when for a joint having some ' θ ' value its ' ϕ ' value is modified by a reasonably large value, some amount of unwanted twist is introduced because of the transformation. This should be removed by some corresponding inverse transformation. Finding this inverse transformation for neutralizing the unwanted twist is easier if the joint model contains twist motion as one of the degrees of freedom.

It is clear from the above discussion of primitive specification of motion that even for bringing the human body to a desired simple orientation the animator has to perform a laborious task. So there is a need for a higher abstraction of motion specification that would help the animator orient the human body with less work.

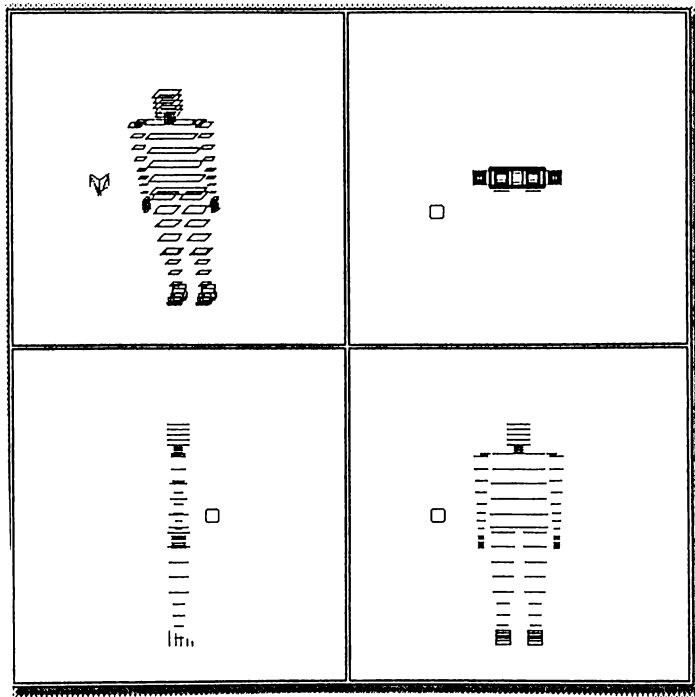
Joint Positioning

As a solution to doing less work while orientation and positioning the body for a key-frame, the system may have the provision for positioning a joint to a desired or goal position in the world coordinate system.

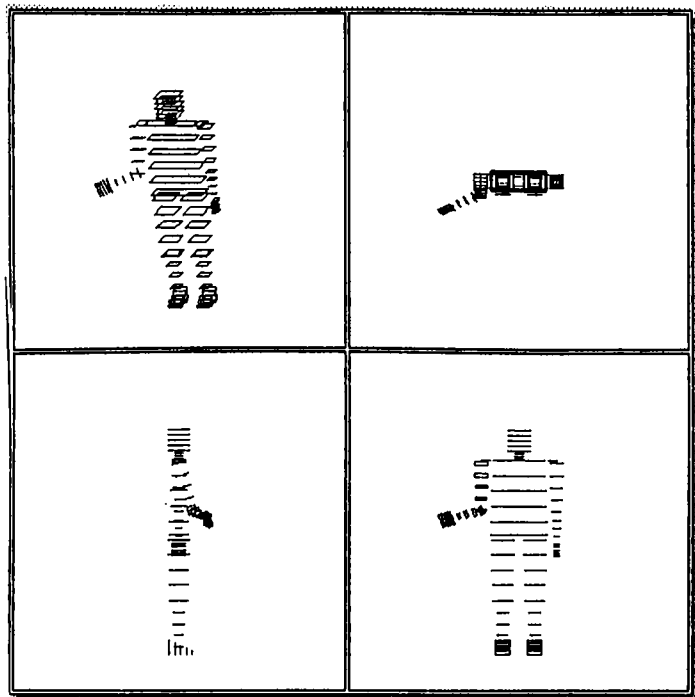
The goal 'g' is defined as a triplet (x_g, y_g, z_g) and the 'joint identifier'. (x_g, y_g, z_g) is specified interactively by the mouse. In AOHM the animator can see the goal position in all the three planes x-y, y-z, and z-x while specifying it (figure 3.3(a)).

After the goal position is specified and the joint is identified (figure 3.3 (a)), the system solves for the related joint parameters and if a solution exists it positions the identified joint to the desired goal position (figure 3.3(b)).

Positioning a joint which is attached to torso or the main body means a translation. Positioning a joint whose parent is attached to torso means that the goal position should lie on the spherical surface spanned by the identified



(a)



(b)

Figure 3.3: Isometric, top, front, and side views. (a) Goal position and the body with unpositioned joint (b) Body with joint positioned.

joint when its parent joint is treated as the center of the sphere. Moreover, the goal position should lie within the joint limits. There is yet another positioning in which the identified joint's grand parent is attached to torso or it is fixed. This means that we have more degrees of freedom in positioning the joint. Examples of such joints are wrists and ankles for our model. This type of joint positioning is more challenging and requires some dedicated methods, especially if joint limits are to be considered.

There are many algorithms for positioning a joint or finding the joint parameters of other related joints for positioning an identified joint. Most algorithms are explored in industrial robotics and thus do not take into account the constraints of joint limits which is a important issue in our case.

When joint limits are ignored and a joint is identified to be placed at a goal (x_g, y_g, z_g) , the values of the joint parameters of other concerned joints (joints whose states need to be modified in order to place the identified joint at the desired goal position) can be calculated by open chain inverse kinematics. In open chain inverse kinematics a linked body is presented with one end fixed and the angle values for all the joints in the linked body is calculated in order to position the other end to a desired position.

Another step towards simplifying the animator's job would be providing a facility of positioning multiple joints to respective multiple goal positions. Usually not all the identified joints can be put precisely to their respective goals, because it is difficult for the animator to visualize the modified states of the joints when one identified joint is positioned to its goal. Thus for multiple joint positioning a mechanism for approximating the positioning of the joints should be designed and supported. An approach of weighted goal [2] has been proposed for this case. The concept of weighted goal says that whatever be the mechanism of positioning the joints, care should be taken that the joints having higher goal weights should be approximated with lesser error, i.e. closer to their respective goal. For a detailed discussion the reader can refer to [2] where joint limits are not considered.

A method for positioning a single joint with the constraints of joint limits is implemented in AOHM. We developed this method based on the one discussed

in [23].

To understand the algorithm for positioning the joint (end-effector) let us take the case of positioning the wrist of a 3-D arm (body segment consisting of wrist, elbow, and shoulder joints) (figure 3.4).

The shoulder is taken as the origin of the coordinate system, and let the positions of the elbow and wrist be 'e' and 'W', respectively (figure 3.4).

The arm has four degrees of freedom, two each for the shoulder and elbow joint, and the goal has three parameters. There is one degree of redundancy. Thus there is no unique solution rather a set of solutions if there exists any at all.

In figure 3.5 it can be seen that the problem is actually to find a value for the elbow angle ϵ and a value for the angle ϕ , such that the states of shoulder and elbow joints are within their joint limits. The angle ϕ can be defined as the orientation of the plane formed by O, e, and W w.r.t. the **x**-axis when the **z**-axis is taken as coincident to the imaginary line joining O and W. It is further assumed that the axes of the coordinate system originated at shoulder joint have such orientations that when the imaginary line joining O and W i.e. OW is transformed with T_w^{-1} to coincide with the **z**-axis of the coordinate system of the shoulder joint, the two systems should coincide. Finding the ϵ and ϕ angles suffices because shoulder joint O is inherently fixed and the wrist joint is fixed because of the goal constraint.

Basic Algorithm :

- 1. First check if $\|O\vec{W}\| < (\text{sum of the upper and lower arm lengths})$.
- 2. Find elbow E angle ϵ (elbow's θ angle) and check it against the joint limits of the joint elbow.
- 3. Find an expression for CIR(ϕ), a vector function that gives a circle on which the elbow is constrained to lie by the shoulder position O and the goal position W of the wrist.
- 4. Determine the arc ARC1(ϕ) on CIR(ϕ) to which the elbow is restricted

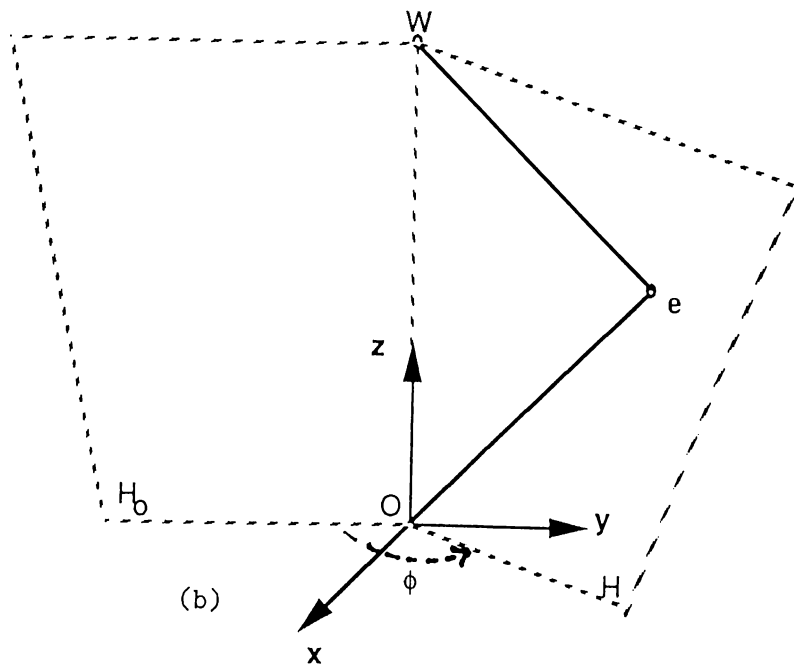
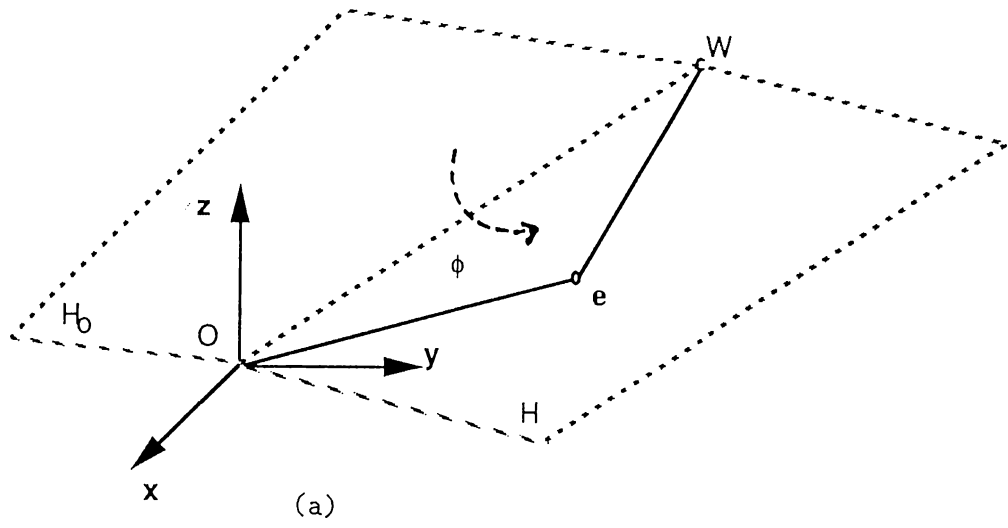


Figure 3.4: (a) Arm model and the plane of the arm (b) Goal line segment \overline{OW} transformed to coincide with the z-axis of shoulder.

by the ϕ limits of elbow joint. Arc $ARC1(\phi)$ means a subset of the domain of ϕ which is permissible by ϕ joint limits of elbow.

- 5. Determine the arc or set of arcs $ARC2(\phi)$ on $CIR(\phi)$ to which the elbow joint E is restricted by the shoulder joint limits.
- 6. Take the intersection of arcs $ARC1(\phi)$ and $ARC2(\phi)$. choose some value of ϕ and calculate the state of shoulder and elbow joints.

Now let us discuss each of these steps in detail:

1. **Goal distance** : Checking whether the goal position $G(x_g, y_g, z_g)$ is very far or within reach, is very simple. From figure 3.5 one can say that,

$$\|W\| < l_u + l_l$$

means within reach, otherwise too far, i.e. no solution possible. $\|W\|$ denotes the magnitude of the vector W from shoulder to the goal as wrist position, l_u and l_l are the lengths of the upper and lower arms respectively.

2. **Elbow θ angle** : Consider figure 3.5, interior elbow angle ϵ , can be obtained from the law of cosines as

$$\cos \epsilon' = (l_u^2 + l_l^2 - \|W\|^2)/(2l_u l_l)$$

Here ϵ denotes the θ angle of the elbow and can be obtained as

$$\epsilon = \pi - \epsilon'$$

The value of ϵ should be checked against the limits of the elbow's θ angle.

3. **Elbow circle**: Here an expression for the parametric description $CIR(\phi)$ of the elbow circle (figure 3.6) is required. It is easy to find an expression for $CIR^o(\phi)$ shown in figure 3.6(a), and then apply a transformation T_w to $CIR^o(\phi)$ to yield

$$CIR(\phi) = T_w * CIR^o(\phi) \quad (1)$$

where T_w represents a transformation matrix that maps z-axis in figure 3.6(a) to the vector \vec{W} in figure 3.6(b). Thus if we apply this transformation to each point on the $CIR^o(\phi)$ we can obtain the corresponding points P_{ci} on $CIR(\phi)$.

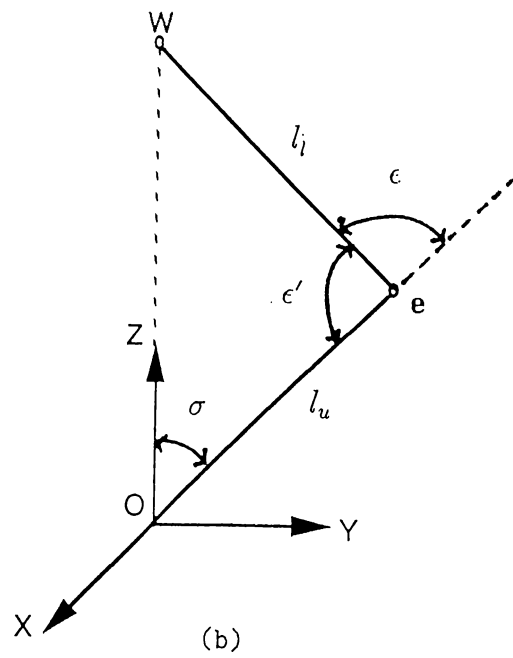
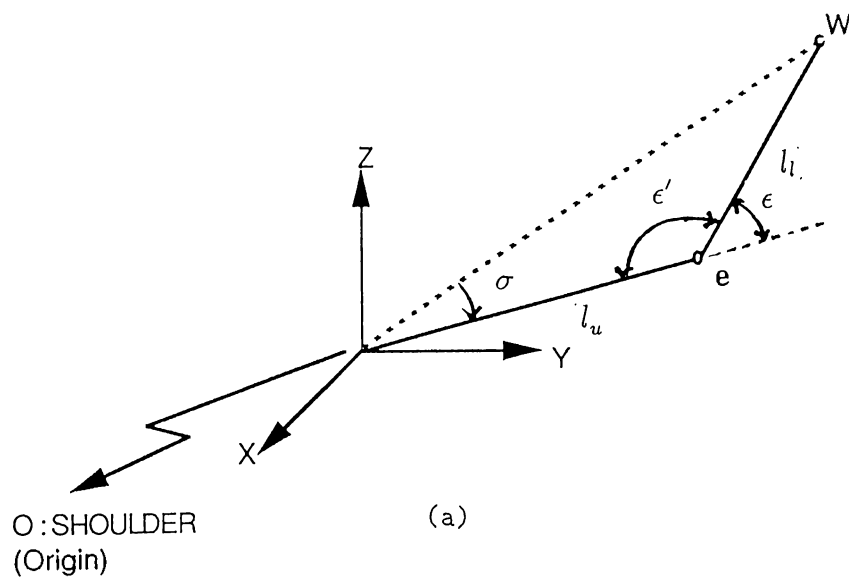


Figure 3.5: Arm triangle (a) Actual (b) Transformed to z-axis of shoulder

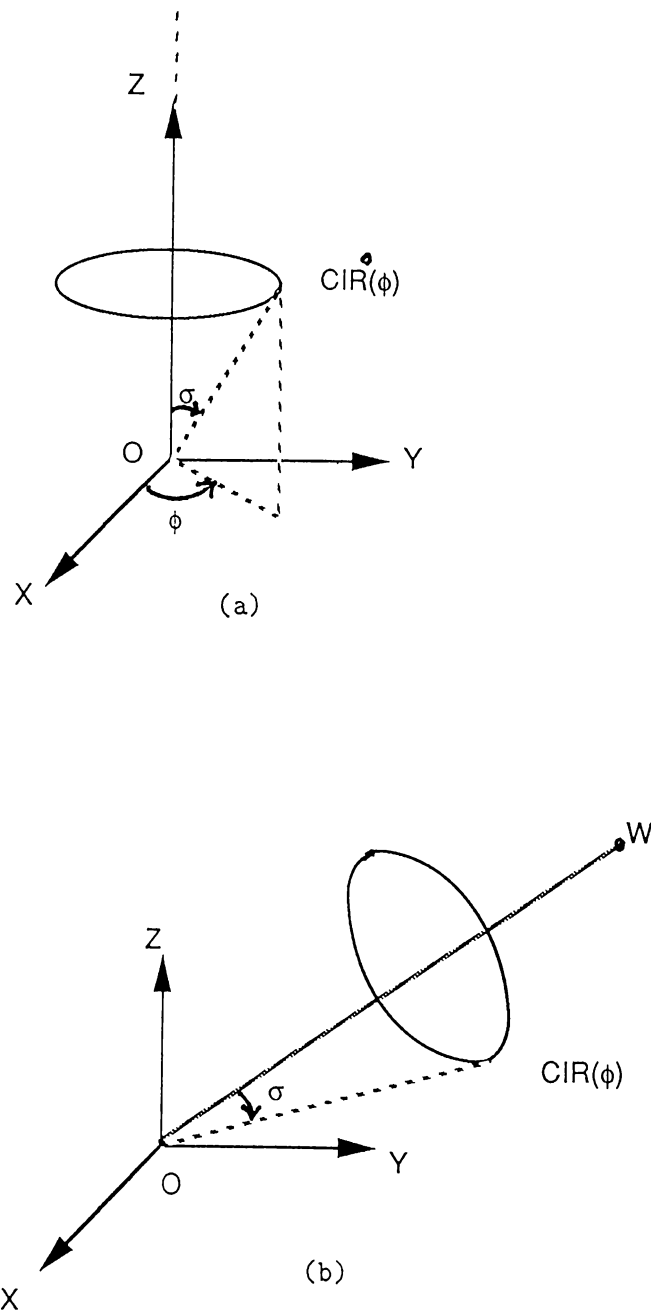


Figure 3.6: Elbow circle with wrist vector \vec{W} on (a) Z-axis (b) Arbitrary original position

To find an expression for $CIR^\circ(\phi)$, σ should be determined. σ can be calculated (figure 3.5):

$$\cos \sigma = (\|W\|^2 + l_u^2 - l_l^2)/(2\|W\|l_u) \quad (2)$$

A sphere of radius l_u around an origin can be represented as:

$$S(\theta, \phi) = l_u \begin{pmatrix} \sin \theta \cos \phi \\ \sin \theta \sin \phi \\ \cos \theta \end{pmatrix} \quad (3)$$

Putting $\theta = \sigma$ in equation 3, we get $CIR^\circ(\phi)$ as:

$$CIR^\circ(\phi) = l_u \begin{pmatrix} \sin \sigma \cos \phi \\ \sin \sigma \sin \phi \\ \cos \sigma \end{pmatrix} \quad (4)$$

Final expression for $CIR(\phi)$ can be obtained with equation 1.

4. Elbow joint ϕ limits: In this step an arc $ARC1(\phi)$ is determined by restricting the ϕ domain of $CIR(\phi)$ with the ϕ limits of the elbow joint. The arc can be obtained simply by considering the upper and lower limit values of the ϕ degree of freedom of the elbow joint, the corresponding ' ϕ ' values in the ϕ domain of $CIR(\phi)$ is noted and $ARC1(\phi)$ is obtained.

5. Shoulder joint limits: Here a subset of the domain of ϕ parameter of the parametric description of the elbow circle, in terms of a set of arcs, $ARC2(\phi)$ is to be determined such that each of the ϕ belonging to this subrange satisfy the joint limit constraints for each of the two degrees of freedom θ and ϕ of the shoulder joint. To understand the issue let us consider the problem geometrically.

In figure 3.7 $CIR(\phi)$ is a circle in 3-D lying on a sphere of radius l_u and center O, the shoulder joint (origin). The joint limits can be represented in spherical coordinates as

$$\begin{aligned} \vec{r}_1 &= (l_u, \theta_{lowlim}, \phi_{lowlim}) \\ \vec{r}_2 &= (l_u, \theta_{lowlim}, \phi_{uplim}) \\ \vec{r}_3 &= (l_u, \theta_{uplim}, \phi_{uplim}) \\ \vec{r}_4 &= (l_u, \theta_{uplim}, \phi_{lowlim}) \end{aligned}$$

It can be seen (figure 3.7) that the four joint limit vectors of shoulder joint

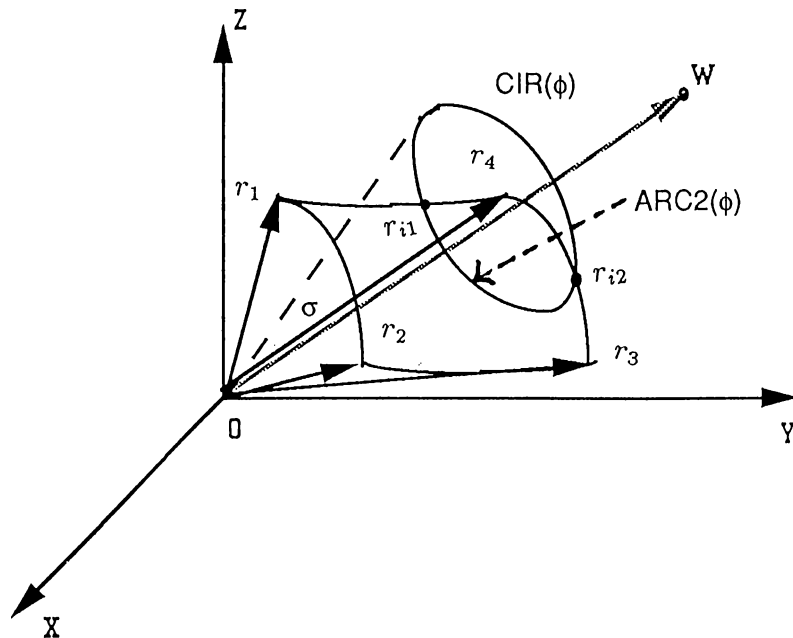


Figure 3.7: Elbow circle and the joint limit boundary.

constitute a spherical polygon Pol-lim $(r_1, r_2, r_3, r_4, r_1)$ on the surface of the sphere of radius l_u and center O. Thus there may exist intersections r_{i1} and r_{i2} between the circle $CIR(\phi)$ and the spherical polygon Pol-lim, i.e. the existence of the arc $ARC2(\phi)$. The problem of finding the intersections can be better understood by considering figure 3.8(a).

What we mean by finding $ARC2(\phi)$, is that, determine the subset of the loci of elbow circle, $CIR(\phi)$ which lies inside the shoulder joint limit, spherical polygon Pol-lim, figure 3.8((a) and (b)). In figure 3.8(b) we see that $ARC2(\phi)$ can be a set of arcs instead of being just a single arc. Figures 3.8 (c) and (d) suggest that in case of no intersection found, a test should be made to find if the whole of the circle is inside the spherical polygon Pol-lim, figure 3.8(d), or totally outside, figure 3.8(c). This test is a very simple one, all one has to do is take any point on $CIR(\phi)$ and test it against each of the extreme values (limits) of each of the shoulder joint's degrees of freedom.

For each of the intersections, r_i 's we know their magnitudes are ' l_u ' and the θ value is ' σ '. Thus, the problem reduces to finding a set of ϕ pairs (ϕ_{min}, ϕ_{max})

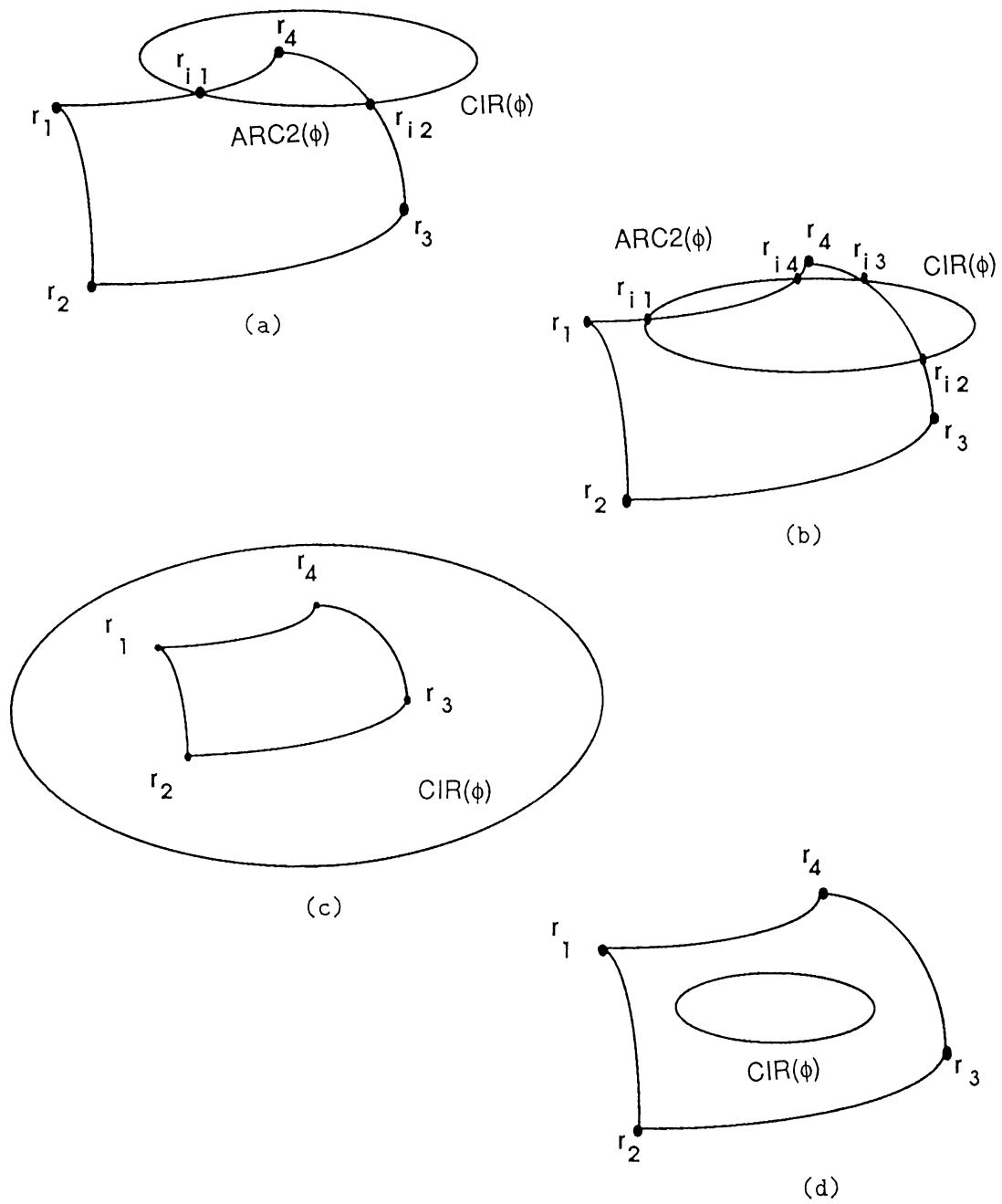


Figure 3.8: Elbow circle and joint limit boundary (a) Two intersections (b) Four intersections (c) No intersection (joint limit boundary inside) (d) No intersection (joint limit boundary outside)

which correspond to the arcs in the set of arcs in $ARC2(\phi)$, i.e.:

$$\phi \text{ domain of } ARC2(\phi) \text{ is } \{(\phi_{1min}, \phi_{1max}), \dots, (\phi_{kmin}, \phi_{kmax})\}$$

The solution of the problem will be simplified if the circle $CIR(\phi)$ and the spherical polygon Pol-lim are transformed by T_w^{-1} , where T_w^{-1} is the transformation that maps the vector \vec{W} to z-axis of the shoulder coordinate system. Thus we have the situation of figure 3.9. For simplicity of notation we shall still refer to these transformed loci as $CIR(\phi)$, $ARC2'(\phi)$, and $Pol - lim'$ are the $ARC2(\phi)$ and Pol-lim.

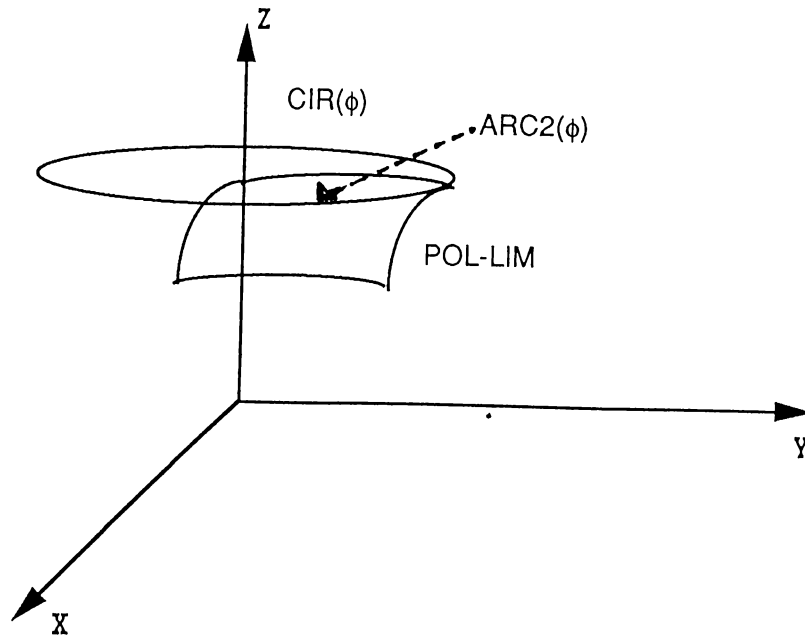


Figure 3.9: Elbow circle and the joint limit boundary, both transformed so that the circle's axis of rotation coincides with the Z-axis of the shoulder.

The problem of finding the intersection between the circle $CIR(\phi)$ and the spherical polygon Pol-lim can be thought of as finding the intersection of the circle $CIR(\phi)$ with each of the arcs $\widehat{r_1 r_2}$, $\widehat{r_2 r_3}$, $\widehat{r_3 r_4}$, $\widehat{r_4 r_1}$, respectively. Thus availability of a method to calculate the intersection of the circle on the sphere with an arc on the same sphere suffices our purpose. we, therefore, state our intersection method designed for this purpose.

Before discussing the method for finding the intersection between the circle $CIR(\phi)$ and the arc $\widehat{r_1 r_2}$, let us see the different cases of how and when the

intersections exist. We have classified the situations as shown in figure 3.10. For analyzing the different situations in figure 3.10 we need to state a definition:

Definition: A point r (vector \vec{r}) on the arc $\widehat{r_1 r_2}$ is inside circle $CIR(\phi)$ if the θ angle of \vec{r} , ' θ ' is less than the constant θ angle, σ , the angle of each point on the loci of the circle $CIR(\phi)$, otherwise we shall say r is outside the circle.

Different cases:

(i) One edge point, r_1 is inside and one edge point, r_2 is outside circle $CIR(\phi)$ so there exists one intersection point, figure 3.10(a).

(ii) Both edges r_1 and r_2 of the arc $\widehat{r_1 r_2}$ are either inside or outside the circle $CIR(\phi)$ so:

(a) \exists no intersection (figure 3.10(b)).

(b) \exists two intersection (figure 3.10 (c)).

(c) \exists two intersection but a tangent point (figure 3.10(d)).

When cases (ii)(a) or (ii)(c) occur we have to do nothing; we simply ignore that arc and skip to the next one in polygon Pol-lim.

In case (ii), we need a mechanism to differentiate between the respective situations (a), (b), and (c).

For cases (i) and (ii)(b) we need to solve for the intersections as follows:

Method : intersections between an arc and a circle

For cases (i) and (ii)(b) a general method can be found which actually always finds two intersection points (vectors) of the two circles $CIR(\phi)$ and $ARCCIR(\alpha)$, where the circle $ARCCIR(\alpha)$ is obtained by extending the arc $\widehat{r_1 r_2}$ from both sides on the spherical surface. Then if case (i) exists simply ignore the unwanted solution (intersection point), otherwise take both the intersections for case (ii)(b). Although this algorithm is general, we shall use the notations in compliance with our purpose.

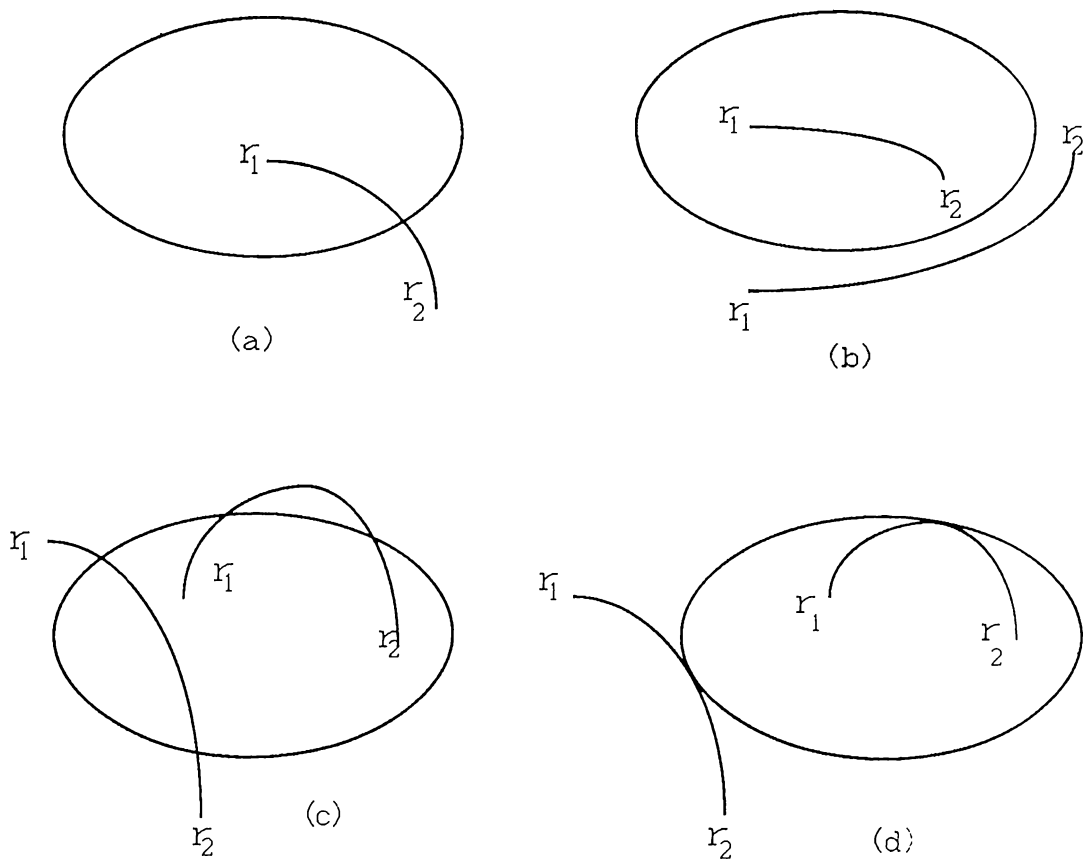


Figure 3.10: Elbow circle and an arc as one of the sides of the joint limit spherical polygon (a) One edge of arc inside and one edge outside the circle and one intersection, (b) Both edges inside or outside and no intersection, (c) Both edges inside or outside and two intersections, (d) Arc tangent to the circle.

We begin with finding a means for differentiation between case (ii)'s (a), (b), and (c). Consider figure 3.11(a), there is a sphere 'S' (i.e. the span of the elbow joint or upper arm centered at shoulder joint and with radius l_u for our purpose). There are three circles shown on the sphere 'S', one is $CIR(\phi)$, the elbow circle. Second is $ARCCIR(\alpha)$, the extension of the arc $\widehat{r_1 r_2}$ and the third circle $EQCIR(\phi)$ is simply the equator of the sphere 'S', taken analogous to the world globe. The third circle can also be thought of as $CIR(\phi)$ with $\sigma = 90^\circ$, i.e. the θ angle of the loci of $CIR(\phi)$ being 90° . The point P_o is one of the intersections of the circle $ARCCIR(\alpha)$ with the equator circle $EQCIR(\phi)$. If we see the sphere S with a viewing vector coincident with the vector \vec{v} shown in figure 3.11 (a), where vector \vec{v} is radial to the sphere S at P_o , we would see the figure 3.11 (a) as figure 3.11 (b).

In figure 3.11 (b) we see circles $CIR(\phi)$, $ARCCIR(\alpha)$, and $EQCIR(\phi)$ as three lines. The vector \vec{n} is the normal to the plane defined by the circle $ARCCIR(\alpha)$, \vec{n} is a unit vector in the direction of $\vec{r}_1 \times \vec{r}_2$ and it can be written as:

$$\vec{n} = (n_x, n_y, n_z) \quad \text{where} \quad \sqrt{n_x^2 + n_y^2 + n_z^2} = 1$$

' γ ' is the angle between the z-axis and the normal \vec{n} . It can be determined as:

$$\cos \gamma = n_z$$

' θ_m ' is the angle sustained by z-axis and the line projection of the circle $ARCCIR(\alpha)$ when measured counterclockwise from z-axis to the projection line. θ_m can be obtained by:

$$\theta_m = \pi/2 - \gamma$$

Now looking at the figure 3.11(b) we have some means to differentiate between cases (ii) (a), (b), and (c):

$$\begin{array}{ll} \text{case (i) or (ii)(b)} & \theta_m < \sigma \\ \text{case (ii)(a)} & \theta_m > \sigma \\ \text{case (ii)(c)} & \theta_m = \sigma \end{array}$$

When we know that case (i) exists, we do not have to do the θ_m test, rather we can directly proceed to find the intersections and discard the unwanted

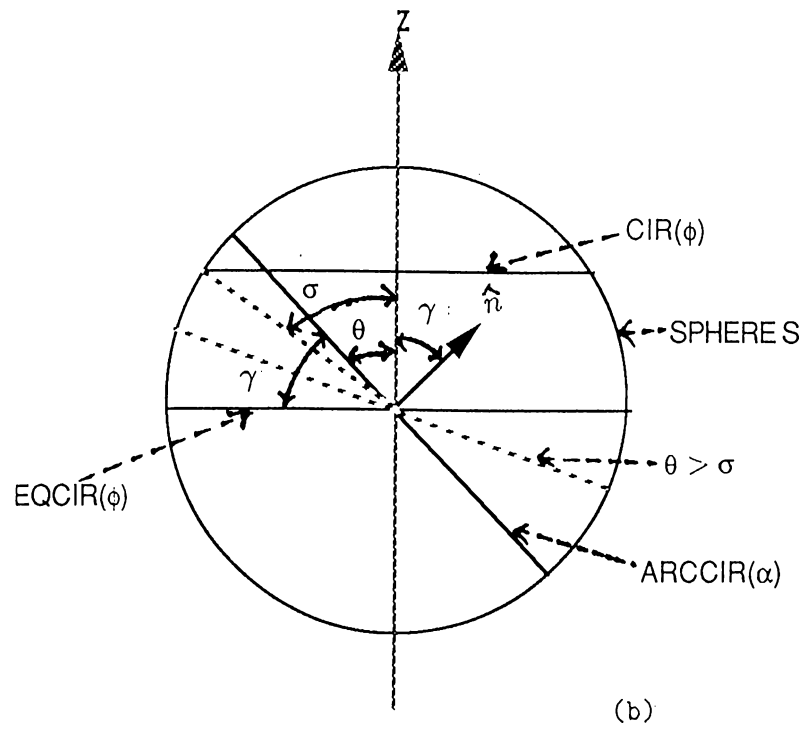
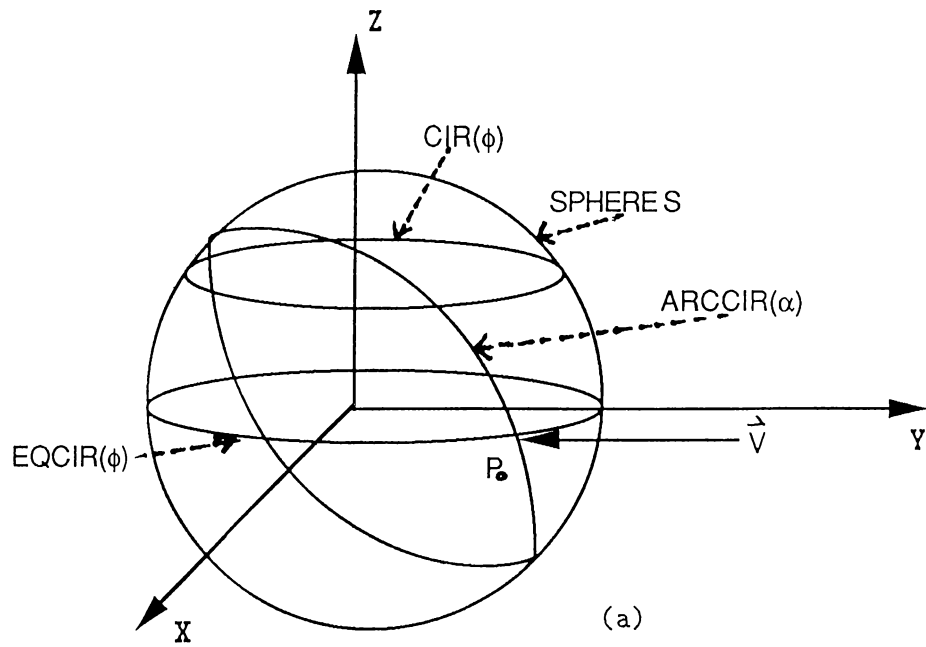


Figure 3.11: Sphere S gives the whole span of the elbow in the absence of shoulder joint limits, elbow circle $CIR(\phi)$, $ARCCIR(\alpha)$ circle obtained by extending the arc and the circle representing the equator of the sphere, (a) Oblique view (b) Viewed from vector \vec{v} .

one. But when case (ii) exists we should do the θ_m test and find out whether we have to simply skip the arc $\widehat{r_1 r_2}$, case (ii)(a) or (c), or proceed to find the intersections, $\theta_m = \sigma$, case (ii)(b).

Method: We first state the known quantities and then the steps of the method.

Given:

- $CIR(\phi)$ with every point on its loci having a constant θ angle value of σ , and l_u vector length.
- An arc $\widehat{r_1 r_2}$ formed by two vectors \vec{r}_1 and \vec{r}_2 :

$$\vec{r}_1(l_u, \theta_1, \phi_1) = (r_{11}, r_{12}, r_{13})^T$$

$$\vec{r}_2(l_u, \theta_2, \phi_2) = (r_{21}, r_{22}, r_{23})^T$$

$$\text{where } r_{i1} = l_u \sin \theta_i \cos \phi_i$$

$$r_{i2} = l_u \sin \theta_i \sin \phi_i$$

$$r_{i3} = l_u \cos \theta_i$$

and each of the θ_i and ϕ_i 's represent the extreme values of the corresponding degree of freedom of the shoulder joint motion.

- Using the given arc $\widehat{r_1 r_2}$ a circle $ARCCIR(\alpha)$ is obtained. Let α be measured from the vector \vec{r}_1 , i.e. the point $ARCCIR(0)$ corresponds to the vector \vec{r}_1 and let α_o be another special value of α such that $ARCCIR(\alpha_o)$ corresponds to the vector \vec{r}_2 .

Now let $\vec{r}(l_u, \theta, \phi)$ be any vector which is a function of ' α ', i.e., it corresponds to a point on $ARCCIR(\alpha)$ with any ' α ' value. Thus for different ' α ' values it will rotate on the circle $ARCCIR(\alpha)$ such that:

$$\vec{r} = \vec{r}_1 \quad \text{for } \alpha = 0$$

$$\vec{r} = \vec{r}_2 \quad \text{for } \alpha = \alpha_o$$

$$\vec{r} = \vec{r}_{i1} \quad \text{for } \alpha = \alpha_1$$

$$\vec{r} = \vec{r}_{i2} \quad \text{for } \alpha = \alpha_2$$

where \vec{r}_{i1} and \vec{r}_{i2} are the two intersections to be calculated.

Here we know that when \vec{r} is equal to \vec{r}_{i1} or \vec{r}_{i2} , out of the three components (r, θ, ϕ) of the vector \vec{r} , two are known i.e. $r = l_u$ and $\theta = \sigma$, since the vector \vec{r} lies on the circle $CIR(\phi)$. Thus only ϕ_{i1} and ϕ_{i2} are to be calculated, i.e., the two ϕ values that correspond to \vec{r}_{i1} and \vec{r}_{i2} .

Steps:

1. Determine α_o by:

$$\alpha_o = \cos^{-1} \left(\frac{\vec{r}_1 \cdot \vec{r}_2}{\|\vec{r}_1\| \|\vec{r}_2\|} \right)$$

2. Determine α_1 and α_2 by:

$$\alpha_1 = \tan^{-1} \left(\frac{k}{\sqrt{1-k_2}} \right) - \tan^{-1} \left(\frac{A}{B} \right)$$

$$\alpha_2 = -\tan^{-1} \left(\frac{k}{\sqrt{1-k_2}} \right) - \tan^{-1} \left(\frac{A}{B} \right)$$

$$\text{where } k = \frac{l_u \cos \sigma}{\sqrt{A^2 + B^2}}$$

$$A = r_{13}$$

$$B = \frac{r_{23} - r_{13} \cos \alpha_o}{\|\sin \alpha_o\|}$$

3. Calculate ϕ_1 and ϕ_2 :

$$\phi_i = \tan^{-1} \left(\frac{r_{12}D - r_{22}E_i}{r_{11}D + r_{21}E_i} \right)$$

$$\text{where } D = \frac{\cos \alpha_o \sin \alpha_o}{\|\sin \alpha_o\|}$$

$$E_i = \frac{\sin \alpha_i}{\|\sin \alpha_o\|}$$

For proofs and derivations see APPENDIX A.

Semi Goal-Directed Approach

This is a new notion introduced in this thesis for motion specification in key-frame animation of articulated bodies like human body. The basic idea is that of the goal-directed animation at the frame level. All the concepts or properties of a goal-directed animation are mapped from film to frame, of course with modifications and simplifications.

In the motion specification hierarchy, semi-goal-directed animation is at a higher level when compared to the joint modification, and the joint positioning. It is at a lower level when compared to the goal directed animation, that is why it is called semi-goal-directed animation.

Semi-goal-directed animation is simply an abstraction of motion specification for key-frame generation and key-frame animation, in which the animator generates the key-frames of the motion sequence by English-like commands. In goal directed approach the whole motion sequence is generated by an English command [7, 11] whereas here only the key-frame is generated.

In this technique, for each of the English commands there is an ordered set of motion specifications at assembly level, most primitive ones like $R(\text{joint-name, angle, axis})$ and $T(\text{disp-x, disp-y, disp-z})$ discussed in section 3.2.1, or at higher level of joint positioning discussed in section 3.2.2. Thus motion planning and controlling in goal directed animation maps to the sequencing of primitive motions. One may argue that the order of primitive motions has no significance as one can modify the θ and ϕ parameters (degrees of freedom of joint motions) of the joints in any order and the end result is the same, so no planning is needed at all. It is right that the order of primitive motion specifications like translation T and rotation R has no order importance but incorporating joint positioning $P(\text{joint-name, } g(g_x, g_y, g_z))$ as one of the members of the set of motion specifications changes the whole theory. This is because joint positioning will yield a different set of joint final states (configurations of joint parameters or orientations of human body) depending on the corresponding set of joint initial states. Especially, translation motion can effect it very much.

A solution for avoiding planning or sequencing the primitive motions would

be, to keep the joint positioning specifications as a subset consisting of rotation specifications only, as joint positioning is also finally implemented as a set of rotation specifications. This is not an acceptable solution, because joint positioning is broken into rotation specifications dynamically, i.e., depending on the states of the joint. The different order of the same set of motion specifications (primitive + joint positioning) may yield different orientations of the human body in the key-frame generations. Thus, planning of the sequence of constituent motion specification of an English command is needed corresponding to the motion planning and controlling in goal directed animation. Here also the planning has to be done at least once like in the case of goal directed animation. From the previous discussion it is clear that parametric goal directed animation is more powerful and versatile than ordinary goal directed animation. The same idea is mapped to the frame level in the semi-goal approach. Before seeing this mapping let's consider the parametric approach in goal directed animation by an example:

- goal directed motion specifications in [7]. The English command is 'walk' and the parameters are 'step-size', 'step-frequency', and 'speed'. The system has default values for each of these parameters, so if the animator does not specify any of these parameters, then the system generates a standard walk (motion sequence frames), where motion planning and control is embedded in the system. The animator can obtain different variations of the walk by assigning different values to one or more of these parameters.

The above concept of parametric goal directed approach is mapped to the parametric semi-goal-directed approach implemented in AOHM. It has English-like commands to specify some desired orientation or desired modification in the state of the human body for the generation of a key-frame. One thing worth noting here is that through the menu-commands (English-Like commands displayed as menu) one can either achieve a desired orientation or a desired modification for an orientation. Achieving desired orientation means the constituent motion specifications are such that whatever be the state of a joint they bring to a particular desired state by primitive operations of rotations, translations, and positioning. Achieving desired modifications means

the joint parameters are changed to a new value by some particular value, that is a desired change is obtained in the state of the joint.

As mentioned, these menu-commands constitute of the following:

$$\textit{Translation} : T = (t_x, t_y, t_z)$$

$$\textit{Rotation} : R = (\textit{joint - name}, \textit{angle}, \textit{axis})$$

$$\textit{Position} : P = (\textit{joint - name}, g(g_x, g_y, g_z))$$

In translation we have three parameters x, y, z displacements. For rotation we have one parameter 'angle', it may be a ϕ angle if 'axis' is 2, or a θ angle if 'axis' is 1. Positioning specification has also three parameters $x, y,$ and z coordinates of the goal position. Thus like in parametric goal directed animation one can change some or all of these parameters present in the definition of motion orientation menu-command and obtain a large variety of similar results. In AOHM, we have provided this facility to the user to modify any of the parameters whenever desired.

As an example, let's take the menu-command 'STEP-RIGHT' from the menu of motion specification in AOHM. STEP-RIGHT consists of the following motion specifications:

```
STEP-RIGHT {
Rotation : (RHIP, 70, 1)
Rotation : (RKNEE, -60, 1)
Rotation : (UPPERTORSO, 15, 1)
Rotation : (LSHOULDER, 30, 1)
Rotation : (RSHOULDER, -30, 1)
Rotation : (LELBOW, 35, 1)
Rotation : (RELBOW, 35, 1)
}
```

We apply this menu-command to the standard initial orientation of the human body in AOHM, figure 3.12 (a) to obtain the orientation of figure 3.12 (b). It can be seen that how simple is the job now, simply with a single command the laborious work of orienting the body to the desired orientation in figure

3.12 (b) is avoided.

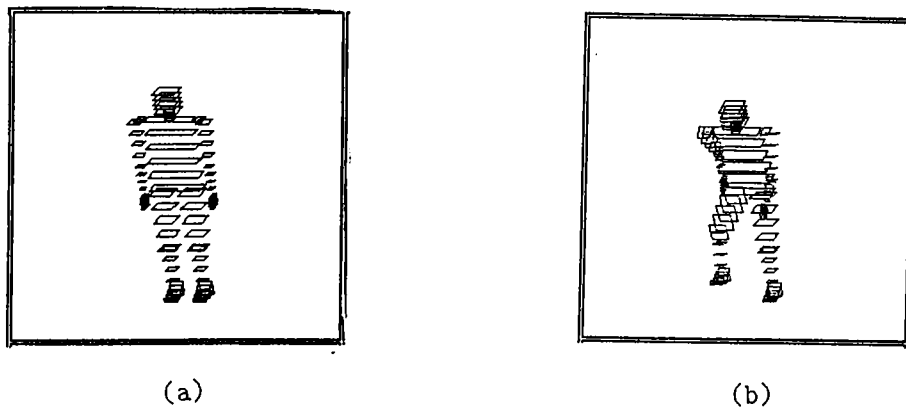


Figure 3.12: Human body orientation (a) Initially (b) After applying the menu-command STEP-RIGHT

Now, if we modify few parameters of 'STEP-RIGHT', for example the angle values in the first two primitive motion specifications, say first angle is changed from '70' to '40' and the second angle is changed from '-60' to '-40', then on applying the same menu-command 'STEP-RIGHT' to the initial orientation of the human body, figure 3.12 (a) we shall get the orientation as in figure 3.13 instead of the orientation of figure 3.12 (b).

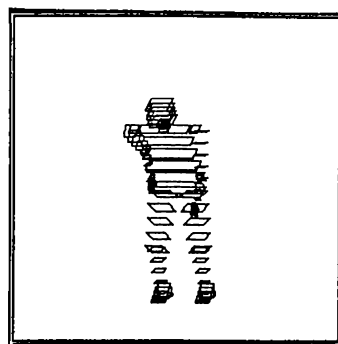


Figure 3.13: Human body orientation after applying the modified menu-command to the initial orientation.

Thus one can achieve a large variety of similar orientations by parametric semi goal directed approach of AOHM.

Another advantage of this semi-goal-directed approach is that, like goal directed approach, no dynamic or kinematic control mechanisms are needed for motion planning and controlling, but semi-goal approach being a modified version of parametric key-frame animation, leaves all the job of motion planning and controlling to the animator, especially in terms of time-stamps. This is a disadvantage of semi-goal approach. Looking to the trade-off between animators job and machine's burden, it is not an unacceptable optimization at all.

To make this notion of motion specifications useful enough, AOHM provides the animator with the facility of defining his/her own pool of menu-commands. This refrains the animator from repeating the difficult tasks once performed. A very good feature of defining own menu-commands in AOHM is that while defining, the animator has all the possible freedom, i.e. animator can use any abstraction of motion specification level, starting from joint modification, assembly level, through joint positioning to including other available menu-commands. There are facilities for deleting some animator defined commands if the animator feels that they are not needed any more.

Goal Directed Approach

This is the highest level of abstraction of the motion specification for generating key-frames in a key-frame animation. In fact, goal directed approach in key-frame animation is almost the same as goal directed animation except that here only the key-frames are generated by motion planning and controlling mechanisms and the inbetweens may be computed by any other algorithm, where as in goal directed animation all the frames are generated by motion planning and controlling mechanism embedded in the system. All the arguments made in section 3.1.1 for goal directed animation are equally valid here and we have included this subsection only for the completeness of our proposed taxonomy for motion specification levels in parametric key-frame animation. According to our information, nobody has so far tried this highest level of abstraction in key-frame animation.

It is hoped that this notion of semi-goal-directed approach will be explored

more and more in the future by us as well as by other researchers.

3.2.2 Inbetweening

Inbetweening is the process of calculating the intermediate frames (in between) between the two animator specified frames (key-frames). In conventional animation this task is performed by an assistant of the animator and in computer animation, computer or machine acting as an assistant performs this job.

As in key-frame animation the animator has to generate each frame of the motion sequence by applying different transformations to the human figure or any figure in general, so it is not practical for the animator to generate all the frames to be displayed during animation. Therefore, the animator specifies the critical ones, the key-frames and inbetweening has to be done by the machine.

Inbetweening can be done over the images, i.e. the inbetweens are generated as images by interpolating the images for the key-frames. This type of inbetweening is termed as image or shape interpolation. This interpolation can be a linear one or may be some complex technique, but whatever be the technique it can never provide satisfactory solution in the deviations between interpolated image [13]. A way of producing better images, especially for articulated bodies like human body is to interpolate parameters of the 2-D projections of the model of the object itself. This technique is a widely used one [9]. In this technique the parameters are the 2-D projections of the degrees of freedom of joint motions. In this technique, inbetweens or intermediate images are generated from the interpolated parameters, and it is termed as parametric or key-transformation inbetweening. Here the quality of images are quite satisfactory with disadvantages like, if the animator modifies a few of the interpolated images, he cannot view the motion from some other different angle unless and until he does corresponding modifications again to that view's motion sequence frames. Moreover, there arises some discontinuities because of interpolating the projections and not the 3-D orientation of the human body.

Better idea would be interpolating the parameters of the 3-D orientation

of the human body itself, thus no information is lost during interpolation. Besides having the advantage of viewing the motion at different angles, it has many other advantages like modification to the interpolated frames can be done in the same manner using all the system facilities as the key frames are generated. Another very important advantage is that path planning during interpolation can be done more realistically as real 3-D space is available, this will be discussed in the next section where the integrating of algorithmic animation to the key-frame animation for achieving more realism is discussed. It is worth mentioning here that this technique as proposed by us, to the best of our informations is being used by nobody, probably because its cost.

Now we shall discuss the above technique of 3-D interpolation implemented in AOHM. This technique consists of two stages. In the first stage, a simple algorithm of linear interpolation over the parameters is applied towards the generation of the inbetween frames. In the second stage, some adjustments to the inbetween frames generated in the first stage are done by making use of some heuristics. At present only one heuristic found by us is implemented in AOHM and we feel that there is a need of some more heuristics to improve the quality of images further. There is a trade-off here between the number of heuristics implemented for quality of inbetween images and the computational cost. Not only the number of heuristics affect the computational cost; rather more effective is the complexity of the heuristic itself.

The two-stage interpolation method is as follows:

Stage 1: In this stage, intermediate frames are generated by a linear interpolation of the joint parameters between the initial and the final joint stages. For example, if for a joint:

initial state is (θ_i, ϕ_i) and

final state is (θ_f, ϕ_f)

where θ and ϕ denote two degrees of freedom of the spherical joint, then the

joint stages for the intermediate frames would be

$$k\text{'th inbetween state} = \left(\theta_i + k \frac{\theta_f - \theta_i}{n + 1}, \phi_i + k \frac{\phi_f - \phi_i}{n + 1} \right) \quad (5)$$

for $k = 1, \dots, n$

where n is the total number of intermediate frames to be calculated between the two initial and final key-frames.

Intermediate frames are generated as transformations with each of the joints having the stages calculated by equation 5. This operation is performed between each pair of given key-frames.

Stage 2: The idea of the second stage in this interpolation method is to decorate the intermediate frames generated in stage 1 with some heuristics so that they look more realistic.

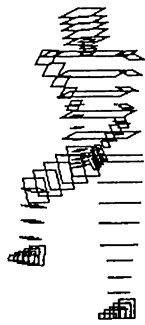
At present AOHM has one simple heuristic for the second stage found by us, which was needed.

The heuristic is that if only one of the end-effectors coordinates are unchanged then for each intermediate frame check whether the coordinate values of that end-effector are the same, if not bring it to that value by translation. If in the two consecutive key-frames (which are to be interpolated), no end-effector or more than one end-effectors have same coordinate values, then check for the coordinate values of the body joint (root joint) in the two key-frames, i.e., whether there is any displacement of the whole body or not. If they are the same then do nothing as a heuristic. If not then translate each of the intermediate frames by :

$$T = \left(x_i + k \frac{x_f - x_i}{n + 1}, y_i + k \frac{y_f - y_i}{n + 1}, z_i + k \frac{z_f - z_i}{n + 1} \right) \quad (6)$$

where (x_i, y_i, z_i) and (x_f, y_f, z_f) are the coordinate values of the body joint (in fact of the whole human figure) in the initial and final key-frames. k is the intermediate frame number, n the total number of intermediate frames to be generated.

Figure 3.14((a) through (e)) gives the idea of the interpolation mechanism supported by AOHM.



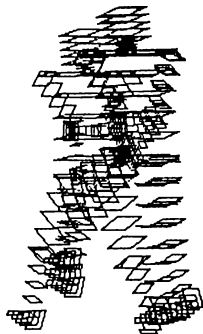
(a)



(b)



(c)



(d)



(e)

Figure 3.14: (a) Initial key-frame, (b) Final key-frame, (c) Two key-frames as initial and final orientations, (d) Key-frames and the inbetweens drawn without heuristic, (e) Key-frames and the inbetweens drawn with the heuristic.

3.2.3 Integrating Algorithmic Animation for Realistic Motion

While interpolating key-frames in *parametric key-frame animation* there is no dynamic control, neither there is any law of nature incorporated in the system, we, therefore, cannot expect from this mechanism to perform very realistically. Thus, when realistic motion is desired or when some motion generated by parametric key-frame animation is found to be unrealistic, then we need some mechanism to add realism the motion.

Motion can be made realistic by integrating *algorithmic animation* into the parametric key-frame animation [13]. The idea of integration is that during interpolation almost all the joint angles are calculated by a normal interpolation method such as the one discussed in the previous section, and few of the critical joint angles are determined from some law of nature attached to it, while generating each of the inbetween. That is to say, use the normal method as long as one gets the acceptable results and if the normal method does not yield acceptable results then use the algorithmic animation technique for that case. Algorithmic animation is used rarely because it is expensive and cannot be used alone practically.

For this kind of integration, a important thing is how can we get the exact trajectories of different joints during the course of different types of human motions. A solution for this is to study the human motion, especially in terms of joint trajectories during the course of a particular motion for different motions depicted by human bodies [27, 28, 29, 41].

As mentioned earlier, algorithmic animation can be applied in different ways, for instance by specifying some function, or mathematical formulae, or procedurally by incorporating dynamics or kinematics. In all of these methods, the desired trajectory is specified. A method for specifying these trajectories would be specifying the loci at some instants, for example at inbetween frame points. We can also define some spline that gives the approximation for the trajectory so that different values of the trajectory loci can be obtained by the system at different occasions. We feel that spline is not a bad choice, as it is

simple and versatile.

To understand how one can integrate algorithmic animation in our approach let us consider the example of stepping of a foot while simulating a walk. Figure

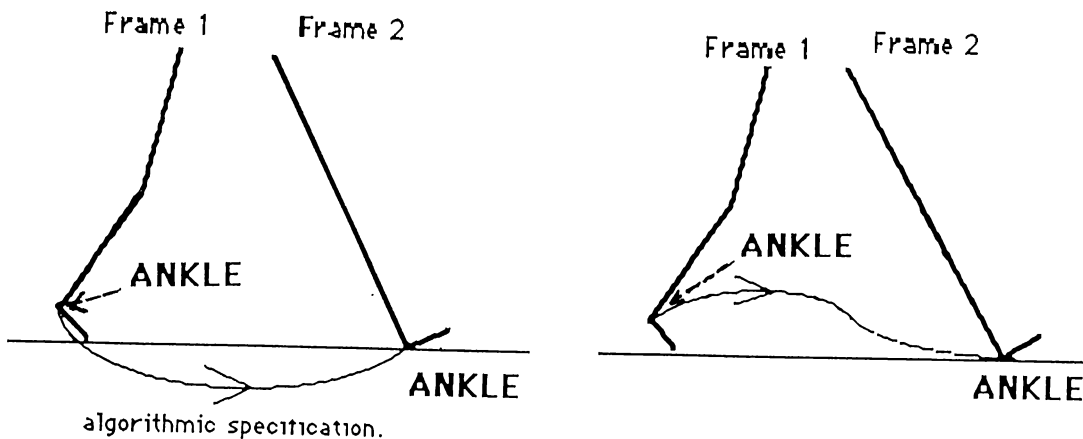


Figure 3.15: (a) Interpolated path of ANKLE in the absence of algorithmic animation, (b) Actual path for the ANKLE,

3.15(a) shows two orientations of a leg as frames 1 and 2, i.e. before and after stepping respectively. The curve shown with an arrow is the path traversed by the ankle while it is interpolated. In figure 3.15(b) the curve shown with an arrow is an approximation of the actual path traversed by the ankle during the stepping of a normal walk. It is obvious from figures 3.15(a) and (b) that curve in figure 3.15(a) is a very bad approximation to the actual path of the ankle. Thus the parametric key frame animation does not seem to work well, as in this case joint angles are interpolated as parameters. Figure 3.16 can be one improvement, here only an extra key frame has been introduced, i.e., frame 2. This is better than the one in figure 3.15(a) but still is not very realistic.

Realism can be improved by increasing the number of key frames but this is not very practical. What we do in our algorithmic approach is that, we provide the animator with the choice of approximating such curvilinear trajectories with cubic splines by specifying a certain number of control points. Together with the control points the animator also needs to specify the slopes of the tangents to the curve at the control points with respect to all x, y, and z. The initial and the final positions of the joint (ankle) is always among the control

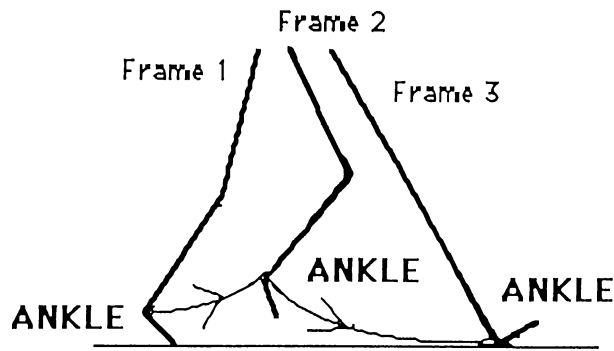


Figure 3.16: Improved interpolated path of ANKLE with additional key-frame.

points as the starting and the terminating control points. In figure 3.17 the animator inserted the control point (x_2, y_2, z_2) and thus the system needs to

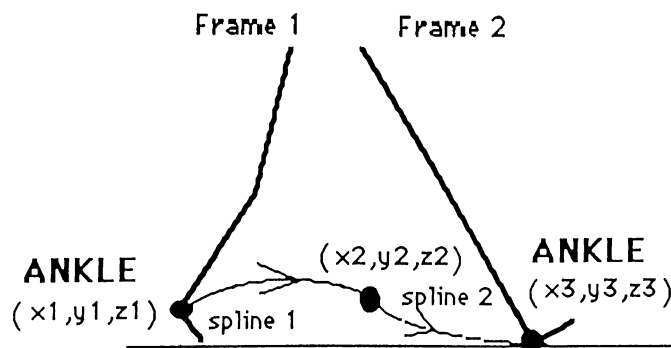


Figure 3.17: Interpolated path of the ANKLE with algorithmic specification by cubic splines.

compute the splines 1 and 2, i.e. the cubic spline between (x_1, y_1, z_1) and (x_2, y_2, z_2) , and between (x_2, y_2, z_2) and (x_3, y_3, z_3) . Here we see that the ankle path approximation is quite good and the accuracy depends on the choice of the inserted control point and the specifications of the slope values. Thus if N control points are inserted by the animator then there will be $N + 1$ cubic splines, and the x , y , and z coordinates of each of the cubic splines will

be of the form:

$$x(t) = at^3 + bt^2 + ct + d \quad (7)$$

$$y(t) = at^3 + bt^2 + ct + d \quad (8)$$

$$z(t) = at^3 + bt^2 + ct + d \quad (9)$$

where 't' is a parameter such that $0 \leq t \leq 1$. Now, since each of the three equations are of the same form, it should suffice to discuss only one. Consider the equation for x coordinate and assume that we are interpolating between control points (x_1, y_1, z_1) and (x_2, y_2, z_2) . Here it is obvious that $x(0) = x_1$ and $x(1) = x_2$. Moreover, since the animator also specifies the slopes of the tangents to the spline at the control points, $x'(0)$ and $x'(1)$ are also known. Thus by simple calculus, the constants a, b, c and d are calculated as follows:

$$a = 2x(0) - 2x(1) + x'(0) + x'(1) \quad (10)$$

$$b = -3x(0) + 3x(1) - 2x'(0) - x'(1) \quad (11)$$

$$c = x'(0) \quad (12)$$

$$d = x(0) \quad (13)$$

After obtaining the equations of the path we calculate the coordinates of the joint (ankle in figure 3.17) by assigning different values to the parameter 't'. The number of different values assigned to the parameter 't' depends on the number of frames required to be produced. Rest of the joint angles are calculated by inverse kinematics.

4. AOHM IMPLEMENTATION

AOHM (Animation of Human Motion) is designed and implemented in C programming language [22] using SUNVIEW¹ [36] on SUN² Workstations running under UNIX³ operating system.

AOHM is a general purpose tool with which wide range of human motions can be generated. The facilities supported for an animator are as follows:

- creation of a new sequence of frames
- modification of a sequence of frames
- reading a sequence of frames from a file
- saving a sequence of frames into a file
- interpolating a sequence of frames
- animating a sequence of frames
- adding a key frame to a sequence of frames
- removing a key frame from a sequence of frames
- preparing a key frame orientation of the body by joint parameter modification
- preparing a key frame orientation of the body by joint positioning

¹SUNVIEW is a software facility for working in the window environment, a registered trademark of Sun Microsystems, Incorporated.

²SUN Workstation is a registered trademark of Sun Microsystems, Incorporated.

³UNIX is a registered trademark of AT&T Bell Laboratories.

- preparing a key frame orientation of the body by menu commands
- creating own menu command

In the following sections some of the important data structures are discussed and a brief comment on the user interface of the AOHM implementation is provided. Those interested in details of the user interface and how to use the tool should see the User Manual in APPENDIX B.

4.1 DATA STRUCTURES

In this section some of the important data structures used in the design and implementation of the human motion animation tool (AOHM) are discussed.

4.1.1 Joint

This structure gives the node definition for representing the joint information in the hierarchical definition of joint tree of the human body.

```
typedef struct Joint {
    char joint-name[NAMESIZE]; /* joint identifier */
    POINT joint; /* joint coordinates */
    double theeta, /*  $\theta$  parameter value */
        phi; /*  $\phi$  parameter value */
    double br-dist; /* absolute distance of 1st child */
    int tot-branch; /* total # of siblings */
    struct Joint *par-joint, /* pointer to parent joint */
        **branch-joint; /* list of sibling pointers */
    SEGMENT *base-seg; /* pointer to segment definition */
    short lim-th-up,lim-th-low, /*  $\theta$  limits */
        lim-ph-up,lim-ph-low; /*  $\phi$  limits */
} JOINT;
```

```

typedef struct Point {
    double x,y,z /* x, y, and z coordinates */
} POINT;

typedef struct Segment {
    char segment-name[NAMESIZE]; /* segment identifier */
    SEG-SHAPE *shape; /* pointer to shape info */
    struct Joint *base-joint; /* pointer to corresponding joint */
} SEGMENT;

typedef struct Segment-shape {
    int shape-info; /* # of SQUARES */
    SQUARE **sq /* pointer to shape definition */
} SEG-SHAPE;

typedef struct Square {
    POINT p1,p2,p3,p4 /* four vertices of square */
} SQUARE;

```

Using the joint node defined above a hierarchy of the joints in the human body representation is obtained (figure 2.4). The tree structure is constructed dynamically on reading a file of human body joint information starting from the root node which is supposed to be an identifier to the whole of the figure. As can be seen from the node definition of the joint, there is no restriction on the number of sibling joints so any articulated figure can be read in. Thus it works for any depth of human body definition, that is, regardless of the number of joints. The values read for the joint coordinate from the file account for the initial orientation of the human body.

A state change in the joint is done by modifying the *theta* and/or *phi* values in the respective joint. Before doing any change in the joint state, the modified value to be obtained is checked against the respective joint limits.

4.1.2 Frame

Frames are represented as a contiguous list of nodes 'TRANSMATRIX', where each of the nodes keeps the transformation information of a joint in that particular frame. Transformation information means the transformation needed to bring a joint coordinate specified as a reference, initially defined in the homogeneous coordinate system, to its designated value required for that particular frame. The nodes also hold information about the state of the joint in that frame so that modifications to the frame can be done easily.

```
typedef struct transmatrix {
    int is-identity; /* joint initial position flag */
    char j-name[NAMESIZE]; /* joint identifier */
    double theeta, /* joint state
        phi;      in the frame */
    MATRIX *mat /* transformation matrix */
} TRANSMATRIX;
```

```
typedef struct matrix {
    double elem[4][4] /* a 4x4 matrix */
} MATRIX;
```

```
TRANSMATRIX **frame; /* definition of the frame */
```

4.1.3 Film

A film or frame sequence for a motion is represented as a doubly linked list of nodes 'FILM-FRAMES', where every node holds the information about a single frame of that motion sequence. The information includes a frame number for the identification of the frame, addresses of the previous and next frame in the motion sequence, and a list of transformations for each of the joints. There is also a flag to differentiate between a key-frame and an interpolated frame.

```
typedef struct film-frames {
```

```

int frame-no; /* frame number */
int key-frame; /* key-frame flag */
struct film-frames *next, /* next frame pointer */
                  *previous; /* previous frame pointer */
TRANSMATRIX **fr-trans; /* frame transformation info */
} FILM-FRAMES;

```

4.1.4 Menu Command

This is one of the most important data structures of the AOHM implementation, since it has been used for supporting our new notion of the abstraction of motion specification in the key-frame generation. It is a contiguous list of nodes, where each of single node stands for a particular ‘menu command’, a command that facilitates the animator to achieve complex orientations of the human body just by an English-like command.

```

typedef struct Menu-oper {
    char com-name[NAMESIZE];
    short tot-oper, /* # of primitive operations */
          sys-flg; /* true indicates tool command and
                  cannot be deleted by user */
    PRIMITIVE-OPER *prim-op; /* list of primitive operations */
} MENU-OPER;

```

```

typedef struct Primitive-oper {
    char j-name[NAMESIZE];
    short oper, /* 1:rotate wrt y, 2:rotate wrt z,
                3:translate, 4:positioning */
          mod-flg, /* parameter modified */
          rel-flg; /* operation relative */
    union p-op-val {
        POINT *pnt; /* pointer to
                    double *rot-ang; parameters */
    } val, *mod-val;
}

```

```
} PRIMITIVE-OPER;
```

A menu command list is defined as:

```
MENU-OPER *m-com-list;
```

which is effectively an array of menu commands. Each menu command 'MENU-OPER' holds how many primitive operations 'PRIMITIVE-OPER' are there in a single command, a list of them, and a flag 'sys-flg' (for each command) in order to differentiate it from the user defined menu command. This is needed to stop the user from deleting a system command. Each of the single primitive operator 'PRIMITIVE-OPER' keeps information such as what kind of operation it is, with 'oper' flag. There are two concepts for any primitive operator, one is relative operation which modifies an angle to a specified value by adjusting the parameters relative to the parameter of the joint state, and the second is the absolute one. This is checked by 'rel-flg'. In AOHM, there is a facility for the animator to use the menu commands in a more versatile manner by changing the parameter values associated with each of the primitive operations of a menu command. We also do not want the user to be able to change the commands in the absolute sense so whenever a parameter of a primitive operation is modified, it is stored in another area '*mod-val'.

There is also a facility for defining user's own menu commands, save them as other menus, and delete them whenever desired. This facility is implemented in its most flexible form, that is one can define a menu command by using any and every type and level of motion specification. For instance one can do joint parameter modification, the lowest abstraction or joint positioning, a higher level or even other predefined menu commands, the highest abstraction implemented in AOHM. This is achieved by keeping a global flag set whenever user is defining a menu command and each of the primitive operation is added to a temporary list of primitive commands until the user tells that he is done with it. This temporary list is a linked list:

```
typedef struct Prim-oper-list {
```

```

    PRIMITIVE-OPER p-oper; /* primitive operation */
    struct Prim-oper-list *next; /* next operation */
} PRIM-OPER-LIST;

```

4.1.5 Coordinate Axes

It is a well known fact from *robotics*, *human body animation* or from the study of any *articulated body* that each of the joints or segments in a hierarchical structure has its own coordinate system, but for doing any processing in the world coordinate system, it is required that the joint or segment definition in its respective coordinate system be reflected to the world coordinate system. Another important issue is that whenever any of the parent or any level of grand parent joint changes its state then the coordinate axes of all the sibling joints have to be modified if their respective state definition has to remain unchanged as desired.

In the above discussion we note that there is a need for storing and keeping a track of the state changes of the coordinate axes of the joints or segments besides keeping the track of state information of the joints themselves. For that we have defined the following nodes:

```

typedef struct axistrans {
    char j-name[NAME_SIZE];
    MATRIX *axis-tr, /* transformation applied to the joint axis */
           *axis-trinv; /* inverse of the axis-tr matrix */
} AXISTRANS;

AXISTRANS **axes-trans; /* list of axes */

```

Here each node has a joint identifier ‘j-name’, and pointers to two matrices. ‘Axis-tr’ gives the composite transformation applied to the axis of the respective joint, to map the joint from the world origin to the final state in a frame under operation. ‘Axis-trinv’ gives the transformation just inverse to the one in ‘axis-tr’.

At the beginning of the program or at each initialization of frame, 'axes-trans' initializes its node transformation matrices so that 'axis-tr' gives the transformation for bringing the respective joint axes from the world coordinate origin to an initial or standard orientation and position. Later at each modification of the joint states, these list nodes are updated accordingly.

Now, since we have not associated an 'axes-trans' list with each of the frame definition, whenever a frame is loaded, in order to make the loaded frame editable, a valid corresponding 'axes-trans' list is required. This problem can be solved in two ways. One is to associate with each of the frame definitions an axes transformation list. This is unpractical as a lot of memory is wasted in saving a single frame. Second solution (the one implemented in AOHM) is whenever a frame is loaded, to initialize the 'axes-trans' and then adjust it to the corresponding state by using the state information of each of the joint in that frame.

4.2 USER INTERFACE

AOHM's main window is divided into three sections; the topmost window is for messages like film name, frame number under consideration and total frames in the film. The second window is the command window containing the panel buttons. There are three modes for this window; one is the main mode where one can scale the overall working space, set single or multiple views (to be discussed), quit the tool or get to one of the other two modes of frame or film. In the frame mode, a frame editing is done with all the facilities discussed at the beginning of this chapter. In the film mode, film related operations like animate and interpolate are performed. The third window shows the orientations of the human body model while in the frame mode and the motions of the human body model while in the film mode (figure 4.1).

Pop up windows to support different causes and warnings emerge and vanish as the animator proceeds with his motion sequence generations or modifications.

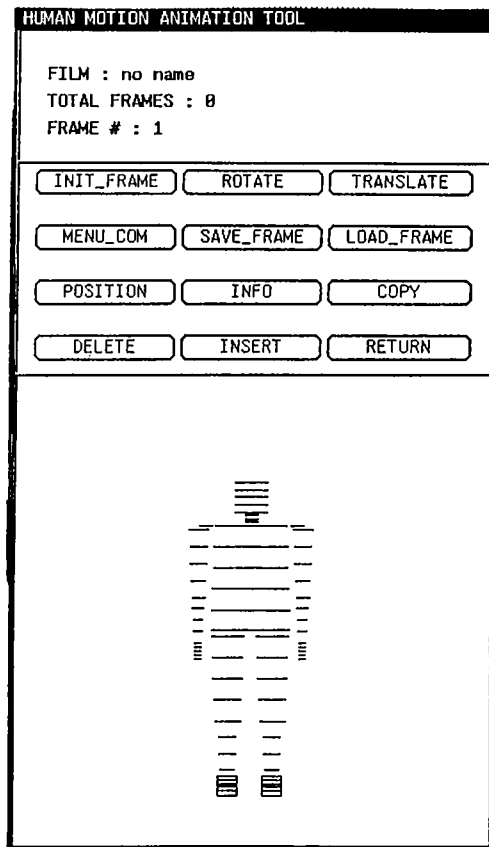


Figure 4.1: Main window of the AOHM tool

One aspect of this user interface is the mechanism developed for entering a 3-D input by the conventional 2-D device 'mouse'. This is required for joint positioning as the animator needs to specify a 3-D desired goal point for positioning a joint. It is also needed in the Algorithmic animation.

The animator is advised to work in the multiple view environment while specifying a 3-D goal point input, (figure 4.2), as in the multiple view, the

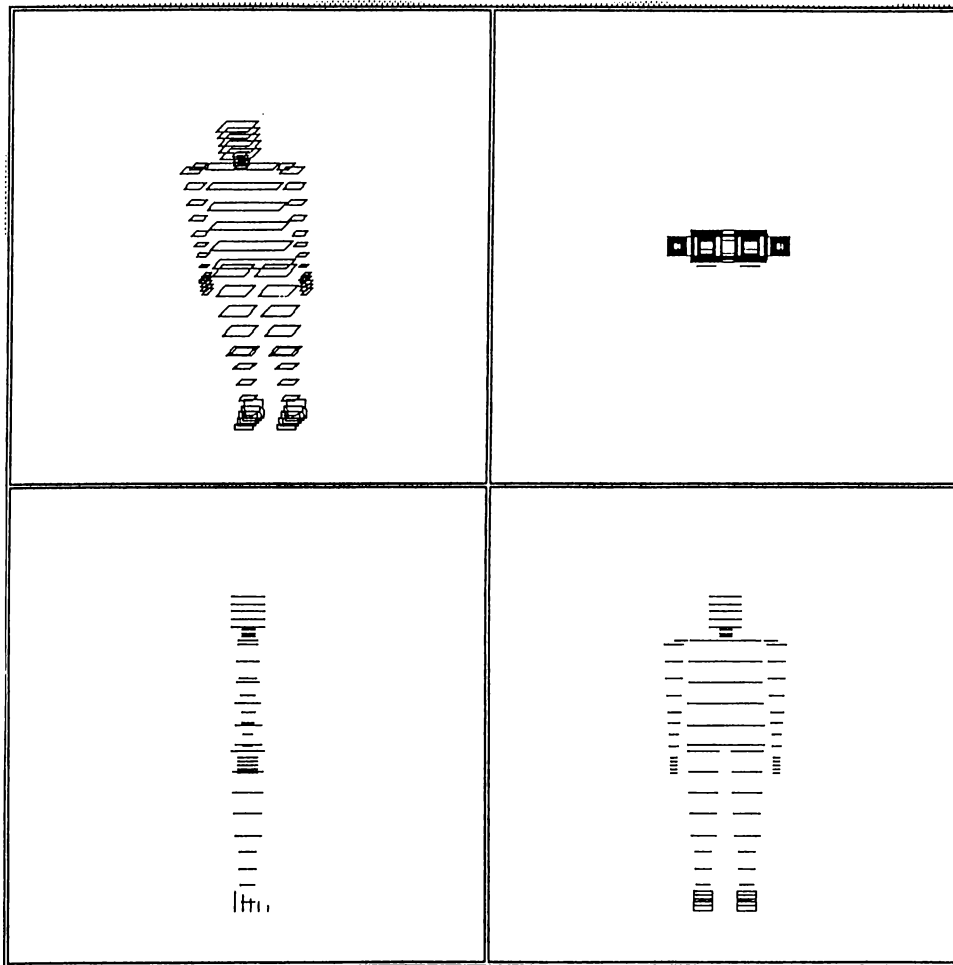


Figure 4.2: Multi-view environment of the AOHM tool

animator can see the top, front, side, and isometric views simultaneously. Thus it is possible to see the movement of the cursor in 3-D.

How is the animator going to move the cursor in the 3-D region with the 2-D device 'mouse'? As a solution, we have defined three modes for the cursor movement, in each mode restricting the cursor to move in one of the three

planes **X-Y**, **Y-Z** or **Z-X**. Following is our interpretation of the mouse button clicks:

MIDDLE-BUTTON down and drag: change of cursor position in the valid plane.

RIGHT-BUTTON down and up: change the plane in the cyclic order of **X-Y**, **Y-Z**, **Z-X**, and **X-Y**, depending on the the state of the plane.

LEFT-BUTTON down and up: present location of the cursor is selected as the desired goal position for some joint to be chosen from the joint name menu. A joint name menu emerges and the animator selects the joint to be positioned.

For more details of the user interface and AOHM, readers are referred to APPENDIX B.

5. FUTURE DIRECTIONS

This research can be regarded as a stepping stone towards the design of a complete, general-purpose, highly interactive, and user friendly human motion animation tool. This is because our approach is modular and we have provided a platform for the motion specification levels. We anticipate the following issues to enhance our research:

- Modeling of human body can be supported interactively by benefitting from the ideas in [9] and [33]. This can also be made menu-driven for high level specifications. Moreover, we feel that if a facility for modeling the scenes surrounding the human figure is supported then its application will increase very much.
- To achieve realistic motion, there is a need for building a joint model which has a joint limit boundary curve very close to the one in actual human body. For this we rely on the research from biomechanics. It is not possible to build a constant model if we want to be very precise as this joint limit boundary curve differs from person to person.
- We feel that for completeness of the work, there is a need to support goal-directed animation. This means motion control and planning mechanisms have to be explored. Building goal directed animation on top of our present tool is simple because of our object-oriented approach. It will be more beneficial if this feature is designed in the way semi goal directed approach is implemented in AOHM. Instead of designing and implementing a complex motion planning and control algorithm, simple algorithms for portions of motion should be implemented, and later

means for combining them should be supported. We suggest a modular approach, in which the simple mechanisms will have all the kinematics and dynamics incorporated in them. Dynamic controls should be used only when extremely needed.

- The ability for showing very long animations is possible if less information is stored for each frame, that is, if we can draw a frame from the very basic information needed for the frame. This is not the case in AOHM. Presently, we keep the transformation information for generating each frame in the motion sequence, as we cannot afford to calculate these while drawing because of limitations of the architecture. This problem can only be solved by a fast machine. Parallelism is the only solution because to obtain the transformation information for each joint, we do not need any communication between the processors.

We think that an implementation purely on a parallel computer will not be an optimal solution; rather ideally distributed system concepts should be used, i.e., the parallel computer should be used as one of the nodes of the distributed environment of the tool. One reason is that when dynamic analysis is run in the parallel computer it should not be disturbed by interactive interruptions. This approach is realized to an extent in [1] where four workstations are used for dynamic analysis. Using distributed environment one can make use of the dedicated machines. The most simple distributed environment would be a workstation and a parallel computer.

6. CONCLUSIONS

A simple, interactive, user-friendly human motion animation tool is discussed. AOHM provides a sound background for researchers in the field of human motion animation, as we started from the very basics of human body animation to some critical issues of this field.

Our proposed taxonomy of the abstractions of motion specification levels in the generation of key-frames is a guideline for the motion specification studies, and it can be considered for achieving better motion specification representations.

The choice of the simple model for human body figure reduced the display and CPU time in the course of animation.

The semi-goal directed approach introduced here simplifies the animator's job and at the same time does not add too much load to the system. The flexibility of being able to define own menu commands restrain the animator from repeating difficult jobs.

The parametric approach in semi-goal directed animation provides the user with the facility of using a menu-command present in the system in the most versatile way.

REFERENCES

- [1] Armstrong, W. W., Green, M. and Lake, R., "Near-Real-Time Control of Human Figure Models," *Proc. Graphics Interface '86 Vision Interface '86*, (1986), pp. 147-151.
- [2] Badler, N. I. et al., "Multi-Dimensional Input Techniques and Articulated Figure Positioning by Multiple Constraints," *Interactive 3D Graphics* (1986), pp. 151-169.
- [3] Badler, N. I., "Animating Human Figures: Perspectives and Directions," *Proc. Graphics Interface '86 Vision Interface '86* (1986), pp. 115-120.
- [4] Badler, N. I., "Design of a Human Movement Representation Incorporating Dynamics," Notes.
- [5] Badler, N. I., "Artificial Intelligence Natural Language and Simulation for Human Animation," Notes.
- [6] Barzel, R. and Barr, A. H., "A Modelling System Based on Dynamic Constraints," *Proc. SIGGRAPH '88* (1988), pp. 179-187.
- [7] Bruderlin, A. and Calvert, T., "Goal Directed, Dynamic Animation of Human Walking," *Proc. SIGGRAPH '89* (1989), pp. 233-242.
- [8] Burtnyk, N. and Wein, M., "Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation," *Communications of the ACM* (1976), pp. 516-521.
- [9] Cachola, D. G. and Schrack, G. F., "Modelling and Animating Three-Dimensional Articulated Figures," *Proc. Graphics Interface '86 Vision Interface '86* (1986), pp. 152-157.

- [10] Computer Graphics and Applications, Institute of Electrical and Electronics Engineers, Vol. 2, No. 9 (Nov 1982).
- [11] Denber, J. M. and Turner, P. M., "A Differential Compiler for Computer Animation," *Proc. SIGGRAPH '86* (1986), pp. 21-27.
- [12] Drewery, K. and Tsotsos, J., "Goal Directed Animation using English Commands," *Proc. Graphics Interface '86 Vision Interface '86* (1986), pp. 131-135.
- [13] Field, D. A., "Mathematical Problems in Solid Modeling," *Geometric Modeling: Algorithms and New Trends*, ed. Farin, G. E., SIAM (1987), pp. 91-108.
- [14] Forest, L. et al., "Integrating Key-Frame Animation and Algorithmic Animation of Articulated Bodies," *Advanced Computer Graphics*, ed. Kunii, T. L. (1986), pp. 263-274.
- [15] Girard, M., "Interactive Design of 3-D Computer- Animated Legged Animal Motion," *Interactive 3D Graphics* (1986), pp. 131-150.
- [16] Girard, M. and Maciejewski, A. A., "Computational Modeling for the Computer Animation of Legged Figures," *Proc. SIGGRAPH '85* (1985), pp. 263-269.
- [17] Goldman, R. N., "The Role of Surface in Solid Modeling," *Geometric Modeling: Algorithms and New Trends*, ed. Farin, G. E., SIAM (1987), pp. 69-90.
- [18] Gourret, J., "Simulation of Object and Human Skin Deformation in a Grasping Task," *Proc. SIGGRAPH '89* (1989), pp. 21-30.
- [19] Grosso, M. R. et al., "Anthropometry for Computer Graphics Human Figures," Technical Report, Dept. of Computer and Information Science, University of Pennsylvania (1988).
- [20] Hearn, D. and Baker, M. P., *Computer Graphics*, Prentice-Hall, Englewood Cliffs, NJ (1986).

- [21] Hodgins, J. K., "Logged Robots on Rough Terrain : Experiments in Adjusting Step Length," Ph.D. Dissertation, Dept. of Computer Science, Carnegie Mellon University (1989).
- [22] Kernighan, B. W. and Ritchie, D. M., *The C Programming Language*, Prentice Hall, Englewood Cliffs, NJ (1978).
- [23] Korien, J. U., "A Geometric Investigation of Reach," Ph.D. Dissertation, MIT Press (1984).
- [24] Laybourne, K., *The Animation Book*, Crown Publishers, New York (1979).
- [25] Lichten, L. and Samek, M., "Integrating Sculptured Surfaces into a Polyhedral Solid Modeling System," *Geometric Modeling: Algorithms and New Trends*, ed. Farin, G. E., SIAM (1987), pp. 109-122.
- [26] Mahmud, S. K. and Özgüç, B., "Human Body Animation," *Proceedings of the Fifth International Symposium on Computer and Information Sciences*, Cappadocia, Turkey (1990), pp. 885-894.
- [27] Martin, D. A. and et al., "Human Upper Limb Dynamics," *Robotics and Autonomous Systems* (1989), pp. 151-163.
- [28] Morasso, P., "Three Dimensional Arm Trajectories," *Biological Cybernetics* (1983), pp. 187-194.
- [29] Morasso, P. and Tagliasco, V., "Analysis of Human Movements: Spatial Localisation with Multiple Perspective Views," *Medical & Biological Engineering & Computing* (1983), pp. 74-82.
- [30] Newman, W. and Sproull, R., *Principles of Interactive Computer Graphics*, McGraw-Hill (1981), second edition.
- [31] Özgüç, B., "Thoughts on User Interface Design for Multi Window Environments," *Proceedings of the Second International Symposium on Computer and Information Sciences*, Istanbul (1987), pp. 477-488.
- [32] Patla, A. E. and Eickmeier, W. E., "Comons: A Computer-Based Movement and Simulation System," *Human Movement Science* (1989), pp. 161-176.

- [33] Reeves, William T. et al., "The Menu Modelling and Animation Environment," *Journal of Visualization and Computer Animation*, Vol. 1 (1990), pp. 33-40.
- [34] Ridsdale, G. et al., "The Interactive Specification of Human Motion," *Proc. Graphics Interface '86 Vision Interface '86* (1986), pp. 121-130.
- [35] Steketee, S. N. and Badler, N. I., "Parametric Keyframe Interpolation Incorporating Kinematic Adjustment and Phrasing Control," *Proc. SIGGRAPH '85* (1985), pp. 255-262.
- [36] Sun Microsystems, *Sun View Programmer's Guide* (1986), Mountain View, CA.
- [37] Sun Microsystems, *UNIX Interface Reference Manual*, Mountain View, CA (1986).
- [38] Thalmann, M. N. and Thalmann, D., *Computer Animation: Theory and Practice* (1986), Springer-Verlag.
- [39] Thompson, David E. and et al., "A Hand Biomechanics Workstation," *Proc. SIGGRAPH '88* (1988), pp. 335-343.
- [40] Tokad, Y. "Mühendislik Sistemlerinin Analizi (Analysis of Engineering Systems)," Part III, Bilkent University, Faculty of Engineering and Science Series No. 1 (1990).
- [41] Velay, J. L. et al., "Elbow Position Sense in Man : Contrasting Results in Matching and Pointing," *Human Movement Science* (1989), pp. 177-193.
- [42] Wang, K., "Computer Graphics Simulation System for Robot Manipulators," *Simulation* (1989), pp. 183-190.
- [43] Wilhelm, J., "Virya - A Motion Control Editor for Kinematic and Dynamic Animation," *Proc. Graphics Interface '86 Vision Interface '86* (1986), pp. 141-146.
- [44] Wilhelm, J. P. and Barsky, B. A., "Using Dynamic Analysis to Animate Articulated Bodies such as Humans and Robots", *Computer-Generated Images*, ed. Thalmann, N. M. and Thalmann, D., Springer-Verlag (1985), pp. 209-229.

- [45] Zeltzer, D., "Towards Integrated View of 3-D Computer Animation," *Computer-Generated Images*, ed. Thalmann, N. M. and Thalmann, D., Springer-Verlag (1985), pp. 230-248.

APPENDICES

A. ARC-CIRCLE INTERSECTIONS

This appendix gives the derivations of the equations (used in Chapter 3) for finding the intersections between a circle and an arc.

We shall basically use the matrix notations for vector operations, namely scalar product and vector product.

Scalar Product:

$$\vec{r}_1 \cdot \vec{r}_2 \longrightarrow \mathbf{r}_1^T \mathbf{r}_2 = [r_{11} \ r_{12} \ r_{13}] \begin{bmatrix} r_{21} \\ r_{22} \\ r_{23} \end{bmatrix}$$

Vector Product:

$$\vec{r}_1 \times \vec{r}_2 \longrightarrow \mathbf{R}_1 \mathbf{r}_2 = \begin{bmatrix} 0 & -r_{13} & r_{12} \\ r_{13} & 0 & -r_{11} \\ -r_{12} & r_{11} & 0 \end{bmatrix} \begin{bmatrix} r_{21} \\ r_{22} \\ r_{23} \end{bmatrix}$$

\mathbf{R}_1 is a 3×3 skew-symmetric matrix and \mathbf{r}_1 is a column matrix, both corresponding to the vector \vec{r}_1 . For the proof of the equivalence of the above matrix notation to the corresponding vector product see [40].

Transformation Matrix $T(\mathbf{n}, \alpha)$: It is 3×3 transformation matrix that when pre-multiplies column matrix \mathbf{v} corresponding to the vector \vec{v} , \mathbf{v} is rotated counterclockwise by α in the plane to which \mathbf{n} is a unit normal. This unit normal actually gives the axis of the rotation of the vector or any object in general. The expression for this transformation matrix is given as [40]:

$$T(\mathbf{n}, \alpha) = \cos \alpha \mathbf{I} + (1 - \cos \alpha) \mathbf{n} \mathbf{n}^T + \sin \alpha \mathbf{N} \tag{1}$$

$$T(\mathbf{n}, 0) = \mathbf{I}$$

Intersection Problem : Now let us define the problem notationally:

Given :

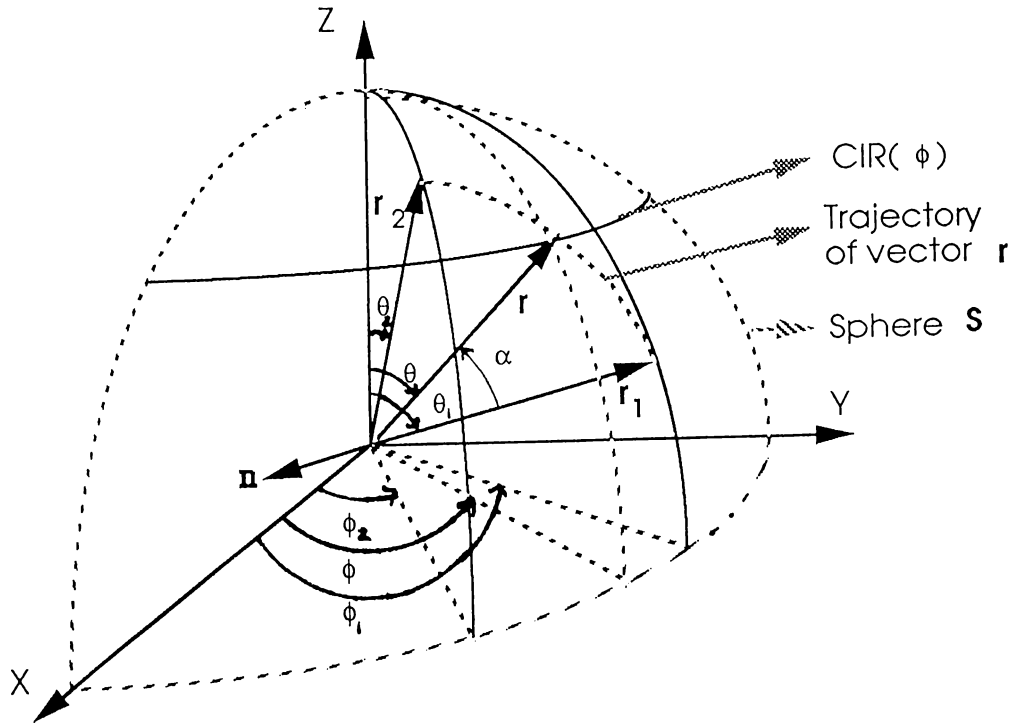


Figure A.1: Arc-circle intersections

A circle $CIR(\phi)$ with its loci vectors having a constant θ angle of ' σ ' and lying on the sphere 'S' of radius 'r', see figure A.1.

An arc $\widehat{r_1 r_2}$ consisting of edge vectors r_1 and r_2 lying on the sphere 'S' of radius 'r', such that:

$$\mathbf{r}_1 = r \begin{bmatrix} \sin \theta_1 \cos \phi_1 \\ \sin \theta_1 \sin \phi_1 \\ \cos \theta_1 \end{bmatrix} \quad (2)$$

$$\mathbf{r}_2 = r \begin{bmatrix} \sin \theta_2 \cos \phi_2 \\ \sin \theta_2 \sin \phi_2 \\ \cos \theta_2 \end{bmatrix} \quad (3)$$

The problem is to find the intersections between the given circle $CIR(\phi)$

and the circle on sphere 'S' obtained by extending the given arc $\widehat{r_1 r_2}$ from both sides until two of the edges meet. Since one of the circles, $CIR(\phi)$ has a constant θ angle loci so the problem reduces to finding only the ϕ values as the magnitudes are also fixed because of the sphere 'S', see figure A.1.

The loci of the given arc is described by the vector \mathbf{r} , such that:

$$\mathbf{r}(\alpha) = f_1(\alpha) \mathbf{r}_1 + f_2(\alpha) \mathbf{r}_2 \quad (4)$$

where $0 \leq \alpha \leq \alpha_o$

$$\|\mathbf{r}(\alpha)\| = \|\mathbf{r}_1\| = \|\mathbf{r}_2\| = r$$

$$\alpha = 0 \Rightarrow \left. \begin{array}{l} f_1(0) = 1 \\ f_2(0) = 0 \end{array} \right\} \mathbf{r} = \mathbf{r}_1$$

$$\alpha = \alpha_o \Rightarrow \left. \begin{array}{l} f_1(\alpha_o) = 0 \\ f_2(\alpha_o) = 1 \end{array} \right\} \mathbf{r} = \mathbf{r}_2$$

The domain of α , $[0, \alpha_o]$ gives the sweep of the vector \mathbf{r} over the given arc.

We can calculate α_o by :

$$\alpha_o = \cos^{-1} \left(\frac{\mathbf{r}_1^T \mathbf{r}_2}{\|\mathbf{r}_1\| \|\mathbf{r}_2\|} \right) \quad (5)$$

The unit vector \mathbf{n} appearing in the transformation matrix expression can be determined as:

$$\mathbf{n} = \frac{1}{\|\bar{\mathbf{n}}\|} \bar{\mathbf{n}} \quad (6)$$

where $\bar{\mathbf{n}}$ is the normal to the plane formed by the two vectors \mathbf{r}_1 and \mathbf{r}_2 . Direction of $\bar{\mathbf{n}}$ is given by:

$$\bar{\mathbf{n}} = \mathbf{R}_1 \mathbf{r}_2 \quad (7)$$

From the definition of the vector \mathbf{r} and the transformation matrix, $T(\mathbf{n}, \alpha)$ we can write:

$$\mathbf{r} = T(\mathbf{n}, \alpha) \mathbf{r}_1 \quad (8)$$

$$= \cos \alpha \mathbf{r}_1 + (1 - \cos \alpha) \mathbf{n} \mathbf{n}^T \mathbf{r}_1 + \sin \alpha \mathbf{N} \mathbf{r}_1 \quad (9)$$

$$= \cos \alpha \mathbf{r}_1 + \sin \alpha \mathbf{N} \mathbf{r}_1 \quad \text{as } \mathbf{n} \perp \mathbf{r}_1$$

$$\begin{aligned}
&= \cos \alpha \mathbf{r}_1 + \sin \alpha \frac{1}{\|\bar{\mathbf{n}}\|} \bar{\mathbf{N}} \mathbf{r}_1 \\
&= \cos \alpha \mathbf{r}_1 + \sin \alpha \frac{-1}{\|\bar{\mathbf{n}}\|} \mathbf{R}_1 \bar{\mathbf{n}} \\
&= \cos \alpha \mathbf{r}_1 + \sin \alpha \frac{-1}{\|\bar{\mathbf{n}}\|} \mathbf{R}_1 (\mathbf{R}_1 \mathbf{r}_2) \\
&= \cos \alpha \mathbf{r}_1 + \sin \alpha \frac{-1}{\|\bar{\mathbf{n}}\|} \mathbf{R}_1 \mathbf{R}_1 \mathbf{r}_2 \\
&= \cos \alpha \mathbf{r}_1 + \sin \alpha \frac{-1}{\|\bar{\mathbf{n}}\|} (\mathbf{r}_1 \mathbf{r}_1^T - (\mathbf{r}_1^T \mathbf{r}_1) \mathbf{I}) \mathbf{r}_2 \\
&= \cos \alpha \mathbf{r}_1 + \sin \alpha \frac{1}{\|\bar{\mathbf{n}}\|} [(\mathbf{r}_1^T \mathbf{r}_1) \mathbf{r}_2 - (\mathbf{r}_1^T \mathbf{r}_2) \mathbf{r}_1] \\
&= \left(\cos \alpha - \frac{\mathbf{r}_1^T \mathbf{r}_2}{\|\bar{\mathbf{n}}\|} \sin \alpha \right) \mathbf{r}_1 + \left(\frac{\|\mathbf{r}_1\|^2}{\|\bar{\mathbf{n}}\|} \sin \alpha \right) \mathbf{r}_2 \quad (10)
\end{aligned}$$

Comparing equations 4 and 10 one can get the expressions for $f_1(\alpha)$ and $f_2(\alpha)$.

Substituting the values of $\mathbf{r}_1^T \mathbf{r}_2$, $\|\bar{\mathbf{n}}\|$ and $\|\mathbf{r}_1\|$ in the equation 10 we get:

$$\mathbf{r} = \left(\cos \alpha - \frac{\cos \alpha_o}{|\sin \alpha_o|} \sin \alpha \right) \mathbf{r}_1 + \left(\frac{1}{|\sin \alpha_o|} \sin \alpha \right) \mathbf{r}_2 \quad (11)$$

Equation 11 actually represents three equations. We can also write the vector \mathbf{r} in terms of its components as:

$$\mathbf{r} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = r \begin{bmatrix} \sin \theta \cos \phi \\ \sin \theta \sin \phi \\ \cos \theta \end{bmatrix} \quad (12)$$

Dividing the second row by the first row in equation 11 and 12 we get:

$$\tan \phi = \frac{r_{12} \left(\cos \alpha - \frac{\cos \alpha_o}{|\sin \alpha_o|} \sin \alpha \right) + r_{22} \frac{\sin \alpha}{|\sin \alpha_o|}}{r_{11} \left(\cos \alpha - \frac{\cos \alpha_o}{|\sin \alpha_o|} \sin \alpha \right) + r_{21} \frac{\sin \alpha}{|\sin \alpha_o|}} \quad (13)$$

From the third row of equation 11 we have:

$$r \cos \theta = r_{13} \left(\cos \alpha - \frac{\cos \alpha_o}{|\sin \alpha_o|} \sin \alpha \right) + r_{23} \frac{\sin \alpha}{|\sin \alpha_o|} \quad (14)$$

Equation 13 gives ' ϕ ' values and equation 14 gives the ' θ ' value for the vector \mathbf{r} corresponding to an ' α ' value. To find the intersections the angle ' θ '

value is known, so we can solve equation 14 for 'α'. Rearranging equation 14 we obtain:

$$r_{13} \cos \alpha + \left(\frac{r_{23}}{|\sin \alpha_o|} - r_{13} \frac{\cos \alpha_o}{|\sin \alpha_o|} \right) \sin \alpha = r \cos \theta \quad (15)$$

$$A \cos \alpha + B \sin \alpha = C \quad (16)$$

$$\begin{aligned} \text{where } A &= r_{13} \\ B &= \left(\frac{r_{23}}{|\sin \alpha_o|} - r_{13} \frac{\cos \alpha_o}{|\sin \alpha_o|} \right) \\ C &= r \cos \theta \end{aligned}$$

To solve a equation of the form 16, a procedure is describe in [40] is the following:

Let

$$A = a \sin \beta \quad (17)$$

$$B = a \cos \beta \quad (18)$$

$$\text{where } a^2 = A^2 + B^2$$

$$\tan \beta = \frac{A}{B}$$

Replacing these expressions of 'A', 'B', and 'C' in equation 16, an expression for 'α' can be established:

$$a \sin \beta \cos \alpha + a \cos \beta \sin \alpha = C$$

$$\Rightarrow a \sin(\beta + \alpha) = C$$

$$\Rightarrow \sin(\beta + \alpha) = \frac{C}{\sqrt{A^2 + B^2}}$$

$$\text{Let } k = \frac{C}{\sqrt{A^2 + B^2}} \quad \text{then } \sin(\beta + \alpha) = k \quad (19)$$

$$\Rightarrow \cos(\beta + \alpha) = \pm \sqrt{1 - k^2} \quad (20)$$

$$\text{Equations 19 \& 20 } \Rightarrow \tan(\beta + \alpha) = \frac{k}{\pm \sqrt{1 - k^2}}$$

$$\Rightarrow \alpha_{1,2} = -\beta + \tan^{-1} \left(\frac{k}{\pm \sqrt{1 - k^2}} \right)$$

$$\Rightarrow \alpha_{1,2} = \pm \tan^{-1} \left(\frac{k}{\sqrt{1 - k^2}} \right) - \tan^{-1} \left(\frac{A}{B} \right) \quad (21)$$

$$(22)$$

Equation 21 gives two values, ' α_1 ' and ' α_2 ' corresponding to the intersection points. Substituting these ' α ' values in equation 13 we obtain the required respective ' ϕ_1 ' and ' ϕ_2 ' values.

B. AOHM USER'S MANUAL

AOHM is used to generate a motion sequence as a computer graphics simulation of the wide range of human body motions. Finally the generated motion sequence can be animated. Using this tool one can also modify a previously generated motion sequence.

To provide the users with a friendly environment, SUNVIEW has been used for creating panels, menus, buttons, etc. [36]. More information about multi window environments can be found in [31]. The modules that comprise the system and the user interface are given in detail in the following two sections.

The Modules

The tool is implemented as several modules containing functions and definitions written in the C programming language. The modules are compiled and linked using the UNIX's *makefile* facility.

The source files are :

```
anim.c          anim-init.c
anim-lib.c      anim-pop.c
a-menu.c        struct-def.h
def.h
```

The definition files include data for the initialization of the program and the motion sequences stored under the directories 'films' and 'rasfilms'.

```
joint.file      seg.file
joint.name      m-com.file

films/         rasfilms/
```

The User Interface

The user interface consists of three subwindows. The first one is a panel that contains message items to give the film name, total number of frames and the frame under consideration information to the animator. The second subwindow contains several command buttons (to be discussed in detail). The third subwindow functions as the graphical display for the tool, as all the animation and human body orientations' display are done in this window (figure B.1).

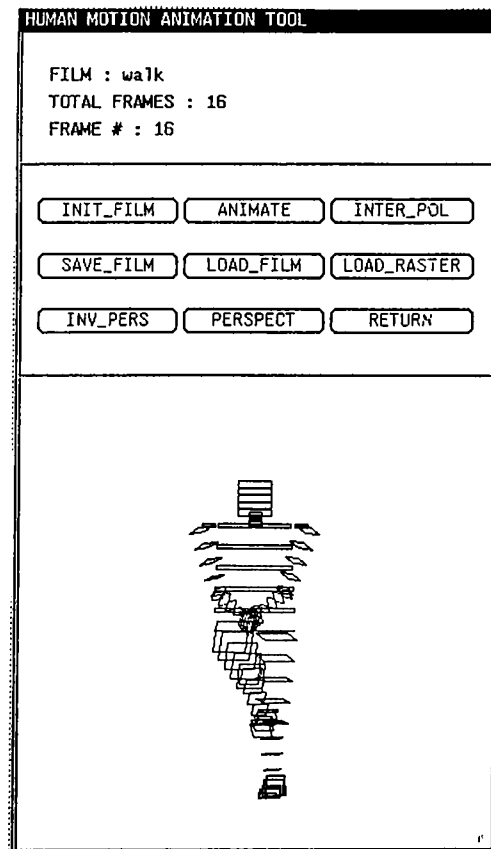


Figure B.1: Main window of AOHM.

The second subwindow which is a panel containing several buttons is the control panel. This control panel has three modes that are discussed in detail in the following subsections.

Main Mode

This is the *default mode* of the this panel and contains six buttons. The function of each button is explained below:

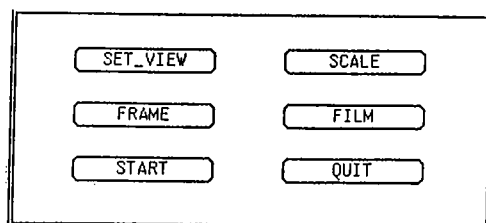


Figure B.2: Button panel in the main mode of AOHM.

- **SET-VIEW** : This is used for setting or resetting the multi view environment. Pressing this button, a pop-up window emerges and informs the user about the state of multi-view and if the user desires, the state can be switched from multi-to single-view and vice versa. In the multi-view state, user is able to see the human body orientation from the top, side, front, and as a isometric view, figure B.3.
- **SCALE** : Pressing this button, a pop-up window, figure B.4 emerges and the user can scale the entire systems, including both the single and multi views, by a percentage factor of the present scale. At the beginning the default value of the scale is taken to be 1.
- **FRAME** : The function of this button is to change the mode of the subwindow from *main mode* to *frame mode*. *frame mode* is discussed in the next subsection, figure B.5.
- **FILM** : The function of this button is to change the mode of the subwindow from *main mode* to *film mode*. *film mode* is discussed in the later subsection, figure B.15.

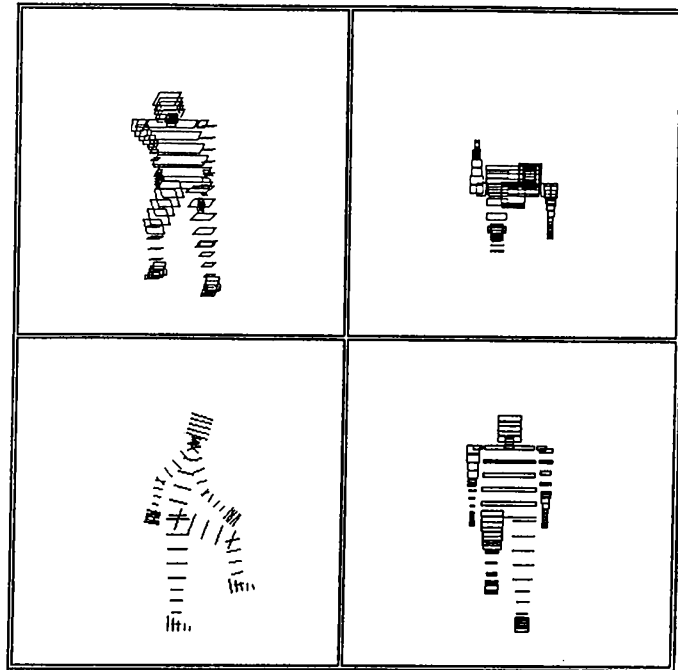


Figure B.3: Multi view environment of AOHM showing isometric, top, side and the front views.

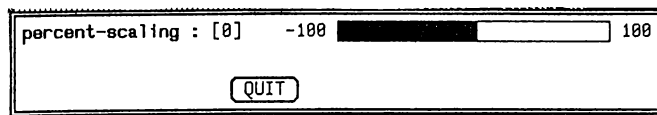


Figure B.4: Scale pop-up window

- **START** : Pressing this button the user can initialize a frame or a film, but it is not a very important one as it is duplicated in the *frame* and *film* modes.
- **QUIT** : This for exiting from the tool and on pressing this the system confirms for the user's desire of exit.

Frame Mode

In the *frame mode* the user can prepare a frame for a motion sequence by using the different facilities supported by the tool AOHM, modify a previously prepared frame, insert or delete a frame by using the button commands explained below:

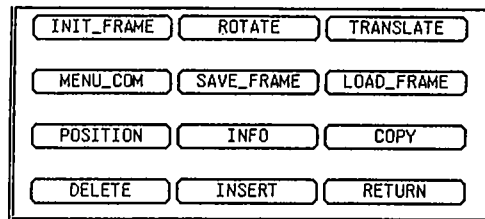


Figure B.5: Button panel in the frame mode.

- **INIT-FRAME** : This button is used to bring the human body orientation to the standard initial orientation shown in the third subwindow of figure B.1, from whatever state of orientation it is in.
- **ROTATE** : Pressing this button a pop-up window emerges, figure B.6. This is for the lowest level of motion specification i.e. joint parameters of the human body is modified to yield different orientations. There are two slider inputs to specify the change in the two degrees of freedom of the joint motion, namely *theta* θ and *phi* ϕ .
- **TRANSLATE** : This button is used for translating the human body in the 3-D space by specifying the **X**, **X**, and **Z** displacement values in the pop-up window, figure B.7.

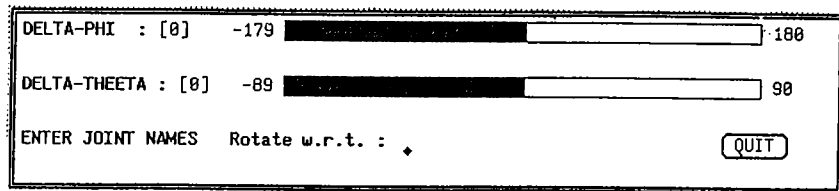


Figure B.6: Rotation pop-up window

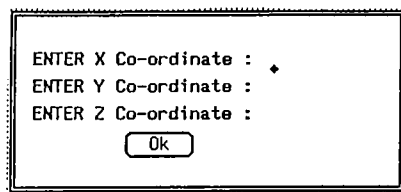


Figure B.7: Translation pop-up window

- MENU-COM : The purpose of this button is to allow the user to enter the mode of highest level of motion specification supported in AOHM. On pressing this button a pop-up panel, figure B.8 emerges containing five buttons explained below:

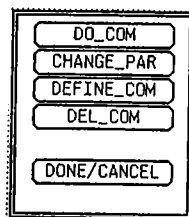


Figure B.8: Pop-up window for menu-command mode.

- DO-COM : Pressing this button a menu of commands appears, and choosing one of these commands the user can achieve different orientations of the human body embedded in the definition of that *menu command*. For example the command *STEP-RIGHT* changes the human body orientation from figure 3.12 (a) to figure 3.12 (b).
- CHANGE-PAR : The function of this button is to provide the user

with a facility to change the values of the parameters of *primitive operations* in *menu command*, to yield a different orientation of human body but close to the one specified by the original *menu command*. Pressing this button a menu of commands appears, and choosing one of these commands another pop-up window, figures B.9 (a) or (b). If the primitive operation is one of the translation or positioning then figure B.9 (a) otherwise figure B.9 (b) for the θ and ϕ primitive rotation operations can be used. At the top of these windows the type of the primitive operation is mentioned followed by the joint identifier. Then the nature of the operation, whether relative or absolute is mentioned. After that if the operation is translation or positioning then three parameters **X**, **Y**, and **Z** coordinates are listed together with their current values which can be modified if the user desires, figure B.9 (a), otherwise the single parameter *phi* ϕ or *theta* θ is displayed, figure B.9 (b).

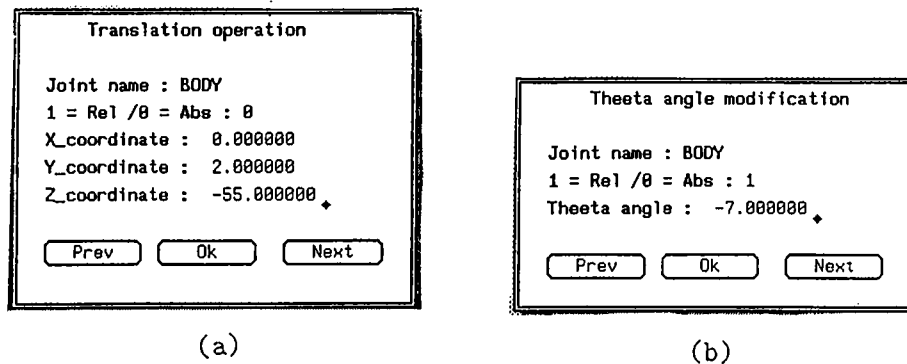


Figure B.9: Pop-up windows to modify the parameter value(s) of (a) translation (b) theta angle rotation

In the last row there are three buttons:

- * Prev : When this button is pressed, the parameter values currently displayed are noted down for any change and the window information switches to another *primitive operation* which is before the one under display in order of their definition sequence in the menu command.
- * Ok : The function of this button is to exit from the change

parameter mode after checking for any change in the parameter values.

- * Next : When this button is pressed, the parameter values currently displayed are noted down for any change and the window information switches to another *primitive operation* which is after the one under display in order of their definition sequence in the menu command.
- DEFINE-COM : Pressing this button, a pop-up window, figure B.10 emerges, informing the user that whatever operation he performs will be recorded as the part of a menu command definition. It also has an operation nature switching mechanism from relative to absolute and vice versa.

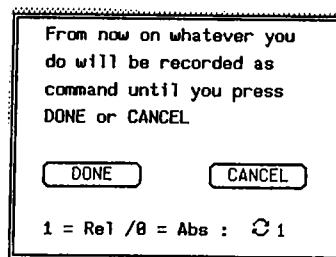


Figure B.10: Pop-up window for helping the user to define a menu-command.

It has two buttons:

- * DONE : Pressing of this button is interpreted as an end to the command definition and the user is prompted for the menu command name.
 - * CANCEL : This button takes the program out of the menu command definition mode by canceling all the operations defined so far.
- DEL-COM : The function of this button is to provide the user with the facility of removing some unwanted menu commands from the pool of user defined menu commands, but the user cannot remove any of the system or tool commands.

- DONE/CANCEL : This simply gets the program out of the menu mode.
- SAVE-FRAME : It saves the current orientation of the human body as a frame in the motion sequence by recording the transformation and state information of each of the joints.
- LOAD-FRAME : It prompts the user with a pop-up window to get the frame number desired to be loaded. If the frame number specified by the user is valid then that frame is loaded for observation and modification, otherwise the user is warned.
- POSITION : This button provides the user with the facility of joint positioning. It lets the user to specify the 3-D goal position (g_x, g_y, g_z) by 2-D device *mouse*. User can see the cursor position in all the four views if in multi-view environment, figure B.11 and the coordinate values in the pop-up window, figure B.12. This is done by defining three modes

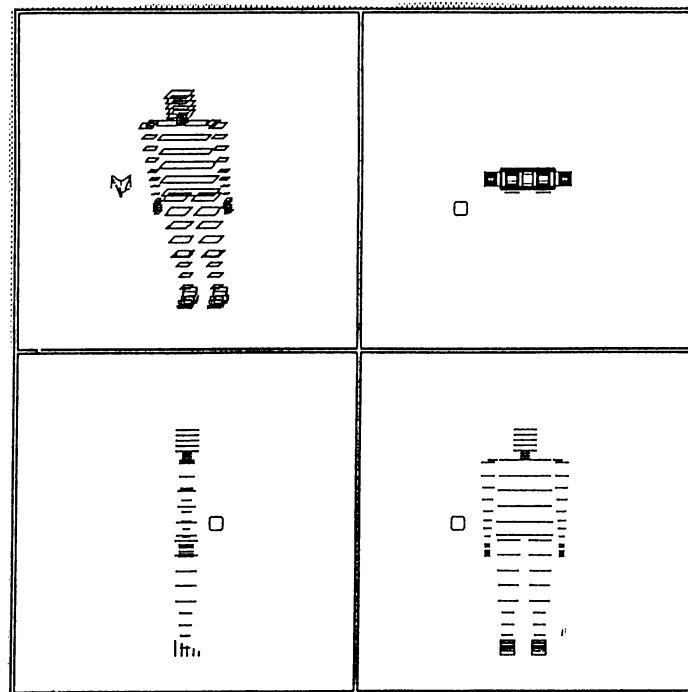


Figure B.11: Cursor positions in the multi-view environment.

for the cursor movement, in each mode restricting the cursor to move in

one of the three planes **X-Y**, **Y-Z** or **Z-X**. Following is our interpretation of the mouse button clicks:

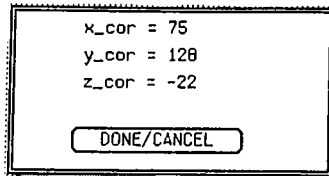


Figure B.12: Pop-up window to give positioning mode of the AOHM and to give the coordinate values.

- When the **RIGHT-BUTTON** is down and up the plane of cursor movement is switched in the cyclic order of **X-Y**, **Y-Z**, **Z-X**, and **X-Y**, depending on the state of the plane.
 - When the **MIDDLE-BUTTON** is down and the mouse is dragged, the coordinate values of any two of the **X**, **Y**, and **Z** is changed depending on which mode of plane is valid. These changes are reflected by showing the coordinate values in a pop-up window, figure B.12, as well as by the cursor movements in the respective subwindows of the multi view window if the user has set the multi view environment, figure B.11. We strongly advise to set the multi view environment while joint positioning because of its ease in doing so.
 - When the **LEFT-BUTTON** is down and up, present location of the cursor is selected as the desired goal position for some joint to be chosen from the joint name menu. A joint name menu emerges and the animator selects the joint to be positioned.
- **INFO** : Pressing this button, a pop-up window, figure B.13 appears. The purpose of this button is to provide the user with the facility of retrieving instantaneous values of the joint's degrees of motion and coordinate values for any orientation of the human body displayed in the third subwindow of the main window of the tool, the canvas.

This pop-up window has three items:

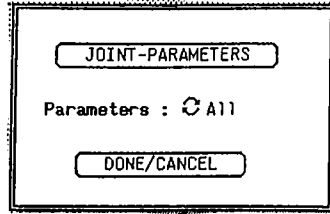


Figure B.13: Pop-up window for joint parameter information retrieving mode.

- **JOINT-PARAMETERS** : Pressing this button, a menu of joint names appears and on choosing a joint from this menu another information pop-up window emerges, figure B.14.

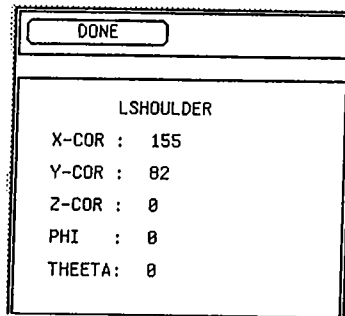


Figure B.14: Information about the joint parameters.

This information pop-up window has two subwindows, first one containing a button **DONE** pressing which the information pop-up window disappears, and the second subwindow is the one that provides the information. In the second subwindow, first the joint identifier is written followed by the joint coordinates and degrees of freedom values, if all the information is to be displayed, or only the joint coordinates, if the coordinates is to be displayed and only the degrees of freedom if only they are to be displayed.

- **Parameters** : It is a panel cycle which switches in the cyclic order of **All**, **XYZ-Cor**, and **PhTh-Ang**, meaning what information is to be displayed, both coordinates and the degrees of freedom, only the joint coordinates or only the joint's degrees of freedom respectively.

- DONE/CANCEL : It merely drives the program out of the information providing mode by making the pop-up window of figure B.13 to disappear.
- COPY : The purpose of this button is to provide the user with the facility of copying one frame information to another frame. The user is prompted for the source and destination frame numbers.
- DELETE : Using this, the user can remove a frame from the motion sequence. It is a very critical operation and so the tool prompts the user for confirmation.
- INSERT : This lets the user to insert a frame between any two frames in a previously defined motion sequence. The user is prompted for the necessary frame numbers.
- RETURN : This button is merely for taking the second subwindow of the tool main window from *frame mode* to the *main mode*.

Film Mode

In the *film mode*, a motion sequence can be loaded, saved, animated, interpolated, etc., by using the button commands discussed below (figure B.15):

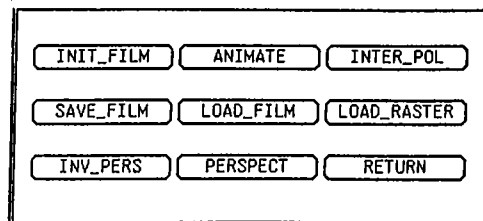


Figure B.15: Button panel in film mode of AOHM.

- INIT-FILM : It initializes the controller so that a new film could be prepared.

- ANIMATE : Pressing this button, the user is able to see *animation*; the motion sequence loaded or prepared in the tool at that instant.
- INTER-POL : The user is prompted for the specification of the number of frames to be calculated as *inbetweens* between the respective frames in the motion sequence.
- SAVE-FILM : This saves the film to the disk on a raster or text file depending on the type of the film currently loaded in the tool, i.e., whether the film is in the form of images or transformations. The user is prompted for the file name.
- LOAD-FILM : Pressing this button, a menu of files appears, and a file from the menu is loaded to the tool.
- LOAD-RASTER : Pressing this button, a menu of raster files appears, and a file from the menu is loaded to the tool.
- INV-PERS : If the motion sequence currently loaded in the tool is viewed *perspectively* then it makes it *parallel projection view*, and if the motion sequence is not in the *perspective view* then it does nothing.
- PERSPECT : If the motion sequence currently loaded in the tool is not viewed *perspectively* then it makes it *perspective*, and if the motion sequence is already in the *perspective view* then it does nothing.
- RETURN : This button takes the second subwindow of the tool main window from *film mode* to the *main mode*.