Representing and Visualizing Articulated Movement

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

By

Matthew Revelle
Bachelor of Science
University of Mary Washington, 2003

Director: Dr. Zoran Durić, Associate Professor
Department of Computer Science

Spring Semester 2009
George Mason University
Fairfax, VA

Dedication

To my mom, thank you for teaching me.
To my friends, thank you for sticking around.

Acknowledgments

In the spring semester of 2008, I had no intention of writing a thesis and was not aware of the interdisciplinary, human movement research being conducted. A meeting with a professor to ensure satisfaction of graduation requirements became a discussion on human movement, languages, and writing a thesis. Without the encouragement, support, and experience of my advisor, Prof. Zoran Durić, Ph.D., this thesis would not have been written. Thank you for introducing me to the topic. Again, ХВАЛА. Writing this thesis required an informed understanding of human movement. Thank you, Dr. Naomi Lynn Gerber, for explaining the anatomical and medical perspectives of movement and helping map mathematical concepts to their anatomical counterpart. The work and support of all students in the Laboratory for the Study and Simulation of Human Movement contributed to the direction and execution of this thesis. In particular, I would like to acknowledge Younhee Kim, Ed Lawson, Michael Sullivan, and Nalini Vishnoi.

Table of Contents

List of Figures

Abstract

REPRESENTING AND VISUALIZING ARTICULATED MOVEMENT

Matthew Revelle, MS

George Mason University, 2009

Thesis Director: Dr. Zoran Durić

This thesis develops a representation and system of operations for articulated limb movement which can be used to describe both specific and general movements. A movement primitive in the representation can be described by the angle, axis, and accelerations of a joint rotation and may be combined with other movement primitives to create a new primitive or a sequence of primitives that simultaneously operate on the joints of a limb. The movement representation in this thesis provides a system for defining symbolic representations of movements. This system includes structured types for a single joint rotation, the sequencing of multiple joint rotations, and a limb model.

To validate the system, an interactive environment with an integrated visualization of a body model has been developed. The environment is embedded in the Clojure programming language, a Lisp dialect that runs on the Java Virtual Machine. The jME scene graph based graphics engine is used to render the body model and visualize movement. The body model is constructed of mesh cylinders for body segments and graph nodes as joints. The environment allows a user to define movements, perform operations on movements, and apply a movement to the body model and view the effect in real time. Two complex movement examples are provided.

Chapter 1: Introduction

Movement generates mobility, completes tasks, and conveys meaning. Understanding and reasoning about movement is important to human health studies, robotics, computer vision, and virtual realities. Aside from its mechanical utility, human movement has many similarities with human language and speech - primitive components are used to construct higher-order components according to a set of rules. There has been significant work done in the fields of linguistics and natural language processing that inspires the study of movement.

This thesis is an attempt at providing an analog of linguistic *words* and *phrases* to movement. In the context of movement, a word is a single rotation of a joint while a phrase is a combination of these joint rotations. By identifying these components of movement and introducing a formal system for their representation and use, it is hoped that greater analyses will be possible.

A system of movement representation needs to support *composition*, *abstraction*, and *precision*. Composition refers to combining or sequencing movements to generate a new movement. Similar to sequencing English words in a sentence, movements may be sequenced to build a higher-order construct. The nature of composition hints at algebraic abstraction: the sentence, "I walk to work," could be abstracted to, "I walk to ____," where an entire set of words and phrases are appropriate values to fill the blank. Movements may similarly be abstracted by leaving an attribute, such as the angle of rotation, undefined. A movement with an undefined angle of rotation can be transformed into many different movements, all with a unique angle of rotation. In order to define a specific movement the movement representation must support precision in the definition of all attributes. When modeling an observed human movement it is necessary to use precise values defining the rotations and velocities of the observed movement.

This thesis focuses on general, articulated limb movement which includes human arm

movements. I define a motion model for joint rotation movement that includes the ability to literally declare a single movement (simple movement) or sequenced movements (super movement). The motion model includes the ability to construct new movements from existing movements and may be extended with new methods of composing movements. A motion model for declaring movements is not useful without a body; so a body model is also defined. This body model includes joints, limb segments, and limbs. Both the motion and body models are presented in Ch. 2.

The author implemented the motion and body models and both are accessible through an interactive, visualized environment. Interaction is important as it facilitates exploration which improves understanding; but, interaction with the models should not be limited to writing and reading symbols and numbers. The ability to write a motion declaration provides a basic human interface with the motion model and three-dimensional visualization of the body model with articulated limbs provides feedback on the effects a motion has on the body model.

The models defined and implemented in this thesis were iteratively developed to meet the goals of composition, abstraction, and precision by recognizing the features of existing movement representations and identifying the gaps in capability.

1.1 Existing Representations

Several representations of movement were reviewed for this thesis: anatomical, dance, and kinematic. Each provides some of the capabilities needed for the goal representation but none support all three.

1.1.1 Anatomical

In anatomy, movements are defined in terms of the neuromusculoskeletal system required by the movement and the effect of the movement [1]. For example, flexion of the elbow joint is a class of movements containing every movement that decreases the joint angle and brings the hand closer to the shoulder [1]. The anatomical representation of movement supports

abstraction, but lacks the ability to represent precise movement.

Movement constraints imposed by physical limitations of the neuromusculoskeletal system are dealt with in anatomy. These constraints may be used to determine the validity of a position or movement and are important to any future work on constrained body models.

1.1.2 Dance

Early movement notations [4] came from dance choreography. In dance, the purpose of a movement representation is to enable communication between dancers and choreographers; the movement notation is similar to music notation. A weakness of dance notations is that the defined movements are ambiguous and rely on context. For example, the acceleration of a movement is expressed with a symbol indicating the dynamics of a particular movement or sequence of movements. Symbols with definitions such as *quickly* or *slow down* are interpreted relatively. Movement composition is the focus in dance notation; precision is achieved through context and oral instruction while abstraction is ignored.

Eshkol-Wachman Movement Notation has been used to describe animal movement [9], martial arts [3], and sign language. The need to increase the discretization density of rotational movement for some of these uses exposes the inherent ambiguity. As the discretization density increases, the notation becomes more complicated and difficult for human understanding.

There have been attempts to augment dance notation with computers [2, 15, 19, 20], but none have resolved the ambiguity inherent in dance notation and its dependence on discretized values for precision.

1.1.3 Kinematic

In robotics, games, and simulations, a rotation representation: quaternions [7,11,12], Euler angles [5], or a transformation matrix [10] are combined with a time duration and interpolation function to produce a sequence of movement frames. Although composition and abstraction may be provided to a limited degree, precise and specific movements designed

for a particular body model are the goal in kinematic representations.

Ad-hoc systems such as these provide a symbolic representation of movement for a specific limb model. There is insignificant support for using the movement definitions in a meaningful way without the limb model.

1.2 Type Syntax

A formal type definition is supplied for each component of the body and motion models. The type notations used in this thesis are inspired by the syntax of the Haskell programming language [6]. The definition of structured types in Haskell provides an explicit description of types with less verbosity than logic notation.

A type name begins with an uppercase and appears on the left-hand of a type declaration. The right-hand side of the declaration contains the type definition, which is separated from the type name by double colons. This, $\mathsf{Foo} :: [\mathsf{Bar}]$, declares a type Foo as being a list of Bars. Lists are denoted with square brackets, [ ]. Tuples are denoted with parenthesis, ( ). Lists may be infinite in length, tuples have an explicit length, which is defined by the number of members. For example, [Real] is a list of any number of Reals, but $(\mathsf{Real}, \mathsf{Real})$ is a tuple of two Reals only. Function types are written as: $\mathsf{Type_1} \times \mathsf{Type_2} \times \ldots \mathsf{Type_n} \rightarrow \mathsf{ResultType}$. There are n types given for a function of n-arity. The type of each argument is separated by $\times$ and the $\rightarrow$ precedes the type of the result.

1.3 Organization of the Thesis

In Ch. 2, movement and body models are introduced and formally defined. An interactive environment for defining and visualizing movement is introduced in Ch. 3. Ch. 4 is a discussion of results and future work and Ch. 5 is the conclusion.

Chapter 2: Symbolic System of Movement

Symbolic systems support abstraction, composition, and rule definitions which in turn provide a framework for modeling an information domain. In this thesis, that domain is movement.

A symbolic representation of movement was developed and implemented for this thesis. This representation requires a body model and a motion model. The body model includes limbs, joints, and orientations, all the components necessary to represent the state of the body. The details of the body model are provided in Sec. 2.1. The motion model describes movements over time in terms of rotations and velocities and is presented in Sec. 2.2.

2.1 Body Model

The body model includes three components: *i*) joints, *ii*) segments, and *iii*) limbs . There is more to a body than limbs, in this thesis we are only concerned with limb movement. Other parts of the body are assumed to be fixed and unaffected by applied movements. The body segments included in the model are: hip, torso, head, right/left upper arms, and right/left forearms.

An important characteristic of the body model borrowed from Eshkol-Wachman Movement Notation [4], is the concept of *light* and *heavy* limb segments. The base of the body is the heaviest segment and each connected segment is considered lighter. A chain of body segments starting at the base and ending at an extremity are ordered from heaviest to lightest. For example, the segment chain of hip, torso, right upper arm, and right forearm are ordered by *weight* from heaviest to lightest. It is valid to say that the hip is both heavier than the torso and right upper arm or that the right forearm is lighter than the right upper

arm. When describing hierarchical relationships between body parts, the concept of light and heavy is applied to both segments and joints.

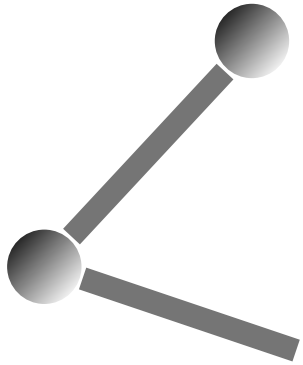

Figure 2.1: *A partial limb. For the upper segment, the joint at the top of the figure is in the heavy role and the joint at the bottom is in the light role.*

The body model exists in a three-dimensional space with a right-handed coordinate system. The up direction is positive y-axis, left is positive x-axis, and straight ahead is positive z-axis.

2.1.1 Joints

Joints serve as roots of individual limb segments and as segment connectors. A shoulder rotation of a human arm results in the rotation of the upper arm, if a joint rotates then the segment attached to it also rotates. In this body model, joints have only a single attribute, orientation, which is defined by a quaternion. When a movement is applied to a joint, the orientation of the joint is updated according to the movement. Joints are the only part of the body model that are directly connected with the motion model.

Each joint may perform one movement at any moment in time.

Definition 2.1: *Type definitions for joints.*

```
Joint :: Orientation
Orientation :: Quaternion
```

2.1.2 Segments

While joints are used to move a limb, segments provide the structure. A segment can be thought of as a bone in a skeletal system. Every segment has a root, or heavy, joint that rotates the segment and light joint that connects one segment to the next.

Formally, segments are rigid bodies in the shape of cylinders. This thesis uses the definition provided by [10], "a rigid body is a collection of particles such that the distance between any two particles remains fixed, regardless of any motions of the body or forces exerted on the body."

Definition 2.2: *Type definitions for segments.*

Segment :: (HeavyJoint, LightJoint, Length)
HeavyJoint :: Joint
LightJoint :: Joint
Length :: Real

2.1.3 Limb

Limbs are chains of segments connected by joints and every limb is attached to a base segment by a joint. In relation to human anatomy, a limb is an entire arm connected to a base segment, the torso, by the shoulder joint. Every joint in a limb may be moved and all the segments lighter than the joint being rotated will move through space. Limb joints can simultaneously rotate, which may occur when a super movement is applied to a limb.

The Limb type is a basic limb representation, and is used in conjunction with the movement types to generate body poses. As this limb model needs to encapsulate information required for geometric transformations there are three essential components: joint orientations, segment lengths, and segment-joint connections.

Limbs may be defined with a set of joints and set of segments. A human arm may be modeled with the three joints: shoulder, elbow, and wrist. The shoulder is the *heavier* of the joints while the elbow is lighter than the shoulder but heavier than the wrist. Fig. 2.1 depicts a partial limb that includes a segment with heavy and light joints.

Each segment is connected to two joints, one joint is the HeavyJoint and the other is the

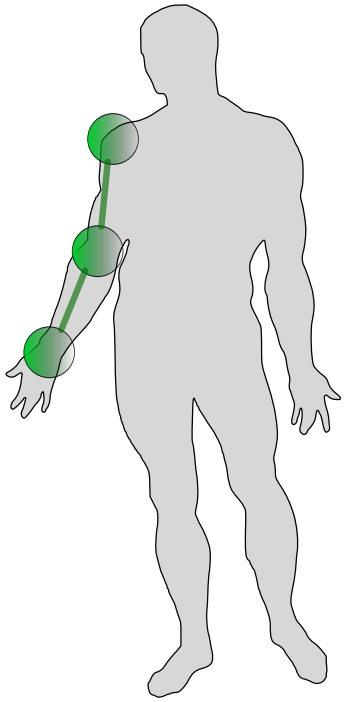

Figure 2.2: *Chain of limb segments connected by joints. Silhouette adapted from [16].*

LightJoint. In a well-defined limb, every joint is *a*) used in two segment definitions, *b*) not used twice in a single segment definition, and *c*) is heavy in one definition and light in the other.

There are two exceptions, the heaviest joint is the root of the limb and is only used in the definition of the heaviest segment as the heavy joint. Similarly, the lightest joint is only used in the definition of the lightest segment where it is the light joint.

Definition 2.3: *Type definitions for limbs.*

Limb :: [Segment]

From the type definition (Definition 2.3), a Limb instance is essentially a set of Joints and Connections. Based on the aforementioned requirements for a well-defined Limb it is possible to create an ordered set of all member Joints, from heaviest to lightest.

2.2 Motion Model

In the motion model, all motions originate from rotations, specifically rotations of joints. Rotations alone do not provide enough enough information to describe a movement, they only describe the trajectory of a movement.

The primitive for symbolic definition of movement is referred to as a *simple movement*. Simple movements are described by the attributes required to represent a joint rotation along a trajectory. It is not required that every attribute of a simple movement be fully defined. This allows an *open* simple movement to represent a subgroup of simple movements that are parameterized by the undefined attributes. Range parameters may also be used for defining movements. Instead of providing a single value for an attribute a range of valid values is used. A movement is considered open if any attribute of the movement is left undefined or it is defined by multiple, potential values. A *closed* movement is a movement with every attribute defined by a single value.

To represent some functional movements, such as eating, it is necessary to define multiple simple movements for multiple joints and a time sequence of their application. All of this information can be stored in a *super movement*.

2.2.1 Rotations

All rotations and orientations are represented by quaternions. A quaternion is an abstraction that is equivalent to an angle and axis of rotation. In terms of an angle, α and axis of rotation, (x, y, z), a quaternion, q, is defined as $q = cos(\frac{\alpha}{2}) + i(xsin(\frac{\alpha}{2})) + j(ysin(\frac{\alpha}{2})) + k(zsin(\frac{\alpha}{2}))$. Quaternions are simple and efficient, and they have algebraic properties that make them ideal for use in this thesis [14, 18].

Formally, quaternions are defined by four numbers: a, b, c, and d. The numbers are broken into a scalar and vector parts, where a is the scalar part, s, and the rest form a vector, $\vec{v}$. From [14], it is shown that quaternion multiplication is a combination of scalar and vector multiplication that is defined in Eq. (2.1).

$$[s_1, \vec{v}_1][s_2, \vec{v}_2] = [(s_1 s_2 - \vec{v}_1 \cdot \vec{v}_2), (s\vec{v}_2 + s_2\vec{v}_1 + \vec{v}_1 \times \vec{v}_2)] \tag{2.1}$$

Spherical rotation is the core of joint movement and quaternions provide the most natural representation. Unit quaternions form a subgroup, S^3, that has a surjection to the group of all rotations, $SO(3)$. As stated in [14], the spherical metric of S^3 is the same as the angular metric of $SO(3)$. For these reasons, quaternion multiplication may be used to combine rotations. The effect is similar to vector addition in Euclidean space which combines translation [14].

Along with clean combination of rotations, quaternions are not subject to gimbal lock and their spherical nature allow for smooth interpolation between orientations. Spherical linear interpolation, or slerp, was first introduced by [14] and may be calculated with Eq. (2.2) where q_1 and q_2 are the quaternions representing the starting orientation and ending orientation and u is an interpolation parameter in the interval [0,1].

$$Slerp(q_1, q_2; u) = \frac{sin(1-u)\theta}{sin\,\theta} q_1 + \frac{sin\,u\theta}{sin\,\theta} q_2 \tag{2.2}$$

2.2.2 Simple Movement

Movements that apply a single rotation to a joint are known as simple movements and include four components: rotation axis, rotation angle, interpolation function, and time duration. Interpolation functions are monotonically increasing with a mapping of $[0, 1] \rightarrow [0, 1]$ where the result is the interpolation value such that $Slerp(q_1, q_2; 0.0)$ is equivalent to the starting orientation and $Slerp(q_1, q_2; 1.0)$ is equivalent to the target orientation.

Definition 2.4: *Type definitions for simple movements and component types.*

SimpleMovement :: (Quaternion, InterpFn, TimeDuration)
InterpFn :: Real → Real
TimeDuration :: Real

2.2.3 Super Movement

Any movement that requires the rotation of multiple joints, e.g. throwing a ball, or requires multiple rotations of a single joint, e.g. hand waving, consists of multiple simple movements. These simple movements may be gathered and sequenced in a single super movement.

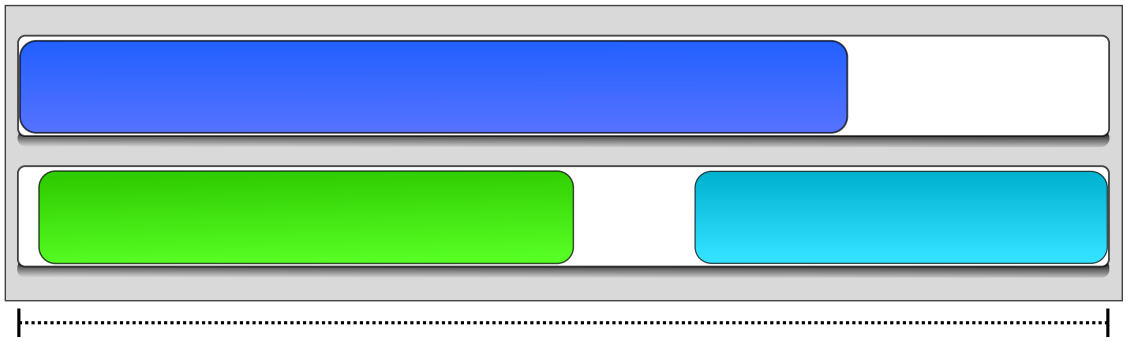

Figure 2.3: *Sequence of movements over time. Three simple movements on two joints define the super movement. The colored blocks are simple movements, and are positioned at a start time. Simple movements may occur simultaneously on different joints.*

Super movements are used to sequence simple movements and maps joints to the simple movement sequences. Each simple movement sequence is a list of 2-tuples of TimeDelay and SimpleMovement instances, the TimeDelay provides the time to wait, relative to the completion of any previous movement, before the application of the corresponding SimpleMovement.

Definition 2.5: *Type definitions for super movements and component types.*

```
SuperMovement :: [(Joint, SimpleMovementSequence)]
SimpleMovementSequence :: [(TimeDelay, SimpleMovement)]
TimeDelay :: Real
```

2.2.4 Operations on Movements

In addition to constructing super movements out of simple movements, the set of operations on movements is extendable. Any function that returns a simple or super movement is a valid movement operator. For example, a function that takes two super movements and

returns a super movement containing the simple movements from both is valid.

The following sections introduce examples of operations on movements.

Combine Rotation

The combination of the rotations of two *SimpleMovement*s is done by quaternion multiplication.

It is important to note that the order of arguments is significant as quaternion multiplication is non-commutative.

Definition 2.6: *Combined rotation.*

$$\mathsf{Quaternion} \times \mathsf{Quaternion} \rightarrow \mathsf{Quaternion}$$

$$q_{combine} = q_2 q_1$$

Combine Interpolation

Two interpolation functions may be combined to form a new interpolation function. The new interpolation function combines the two input functions by taking the original output of the first interpolation function and averaging it with the output of the second interpolation function after mapping the time moment, t, into the domain the first interpolation function.

The resulting function has the domain of the first argument, $[0, duration_1]$, and produces a distance for the given t that is a weighted average of the two functions using a factor α.

Definition 2.7: *Combined rotation.*

$$\mathsf{InterpolationFn} \times \mathsf{InterpolationFn} \rightarrow \mathsf{InterpolationFn}$$

$$interp_{combine}(t; \alpha, duration_1 duration_2) = \alpha interp_1(t) + (1 - \alpha) interp_2(t \tfrac{duration_1}{duration_2})$$

Combine Time Duration

Two movement time durations may be combined in various ways. In this example, the time duration of the result is the weighted average of the time duration of the two arguments.

Definition 2.8: *Combined time duration.*

$$\mathsf{TimeDuration} \times \mathsf{TimeDuration} \rightarrow \mathsf{TimeDuration}$$

$$duration_{combined}(;\alpha) = \alpha duration_1 + (1-\alpha) duration_2$$

Append

The Append operation takes a SuperMovement, SimpleMovement, Joint, and TimeDelay and returns a new SuperMovement with the SimpleMovement appended to the SimpleMovementSequence that corresponds to the given Joint argument. This function provides a way to extend an existing SuperMovement with an additional SimpleMovement.

Definition 2.9: *Appending a movement.*

$$\mathsf{SuperMovement} \times \mathsf{SimpleMovement} \times \mathsf{Joint} \times \mathsf{TimeDelay} \rightarrow \mathsf{SuperMovement}$$

2.3 Data Generation

In order to analyze the effect of a movement on a body, it is necessary to calculate the orientation and position of the body segments and joints as they move. This generated data can be used to answer questions such as, "how is the joint oriented halfway through the movement?" and "what is the position of the forearm after shoulder abduction?" Generation of data is also necessary for visualizing the application of a movement to a body model. When rendering a real time visualization of a body, the orientations of every joint at that frame need to be calculated in order to update the body model.

A series of joint positions and orientations may be sampled during the application of a movement to a limb model. The position and orientation of a limb segment can be found for any moment of time during the application of a movement by calculating the interpolated rotation of the heavy joint for that time instance and applying it to the body model. When updating the limb model, every segment's heavy joint is rotated according to the movement definition being applied.

The joint orientations and positions for a sample can be calculated in a recursive fashion,

see Eq. (2.4) and Eq. (2.5). Similarly, the entire series of movement data may be viewed as a recursive function, see Eq. (2.3).

$$\left\{ q_{t,j} = q_{t-1,j} q_{\Delta_{t-1,t,j}} \mid j \in Joints \right\} \tag{2.3}$$

The function defined in Eq. (2.3) maps the orientations of all limb joints at time t to their orientation at $t+1$. The change in a joint's rotation, q_Δ, is found by spherical linear interpolation [14], see Eq. (2.2). It is necessary to find the orientation of joints relative to the limb's base segment, see Eq. (2.5), in order to compute the segment vector, $\vec{s}_{i-1,i}$, between joint j_{i-1} to joint j_i.

$$position(j_i) = \begin{cases} \vec{x}_0 & \text{if } i = 0 \\ position(j_{i-1}) + \begin{pmatrix} 0 \\ s_{i-1,i} \\ 0 \end{pmatrix} & \text{if } i > 0 \end{cases} \tag{2.4}$$

$$orientation(j_i) = \begin{cases} q_0 & \text{if } i = 0 \\ orientation(j_{i-1}) q_i & \text{if } i > 0 \end{cases} \tag{2.5}$$

Once relative joint orientations for a moment in time have been applied, positions of the limb joints relative to the frame of the limb's base segment can be calculated. The position of every joint is defined with respect to the preceding joint in the limb and the segment that connects them, as shown in Eq. (2.4). Segments are represented by a length, s, and are aligned to the y-axis of the joint (see Sec. 2.1.3).

Chapter 3: Software Implementation

An interactive environment, named Motive, comprising the symbolic system of movement and associated models described in Ch. 2. has been implemented for this thesis in the form of an interactive environment for defining movements and visualizing their application to a body model. The environment may be used to define and manipulate notated movement and apply movements to a three-dimensional mesh model for visualization.

Motive is written in the Clojure programming language, a dialect of Lisp that runs atop the Java Virtual Machine (JVM). Lisp has a history of being used to host interactive environments and domain-specific languages [13,17] and the JVM has several three-dimensional graphics libraries; as a modern Lisp dialect that runs on the JVM, Clojure is an excellent host for Motive. An interactive prompt is used to read movement definitions, move the body model, and access all standard Clojure functionality.

3.1 Definition

All movement definition and manipulation is done using Clojure data structures, functions, and macros. The current implementation is fully functional.

A quaternion representing a 45° (approximately 0.7854 radians) rotation about the x-axis is constructed with `(quat 0.7854 (axis -1 0 0))`. Note that angles in definitions are represented as radians, not degrees. Along with a quaternion, a simple movement definition includes an interpolation function. While it is possible to manually define a function that meets the requirements of the interpolation function given in Sec. 2.2.2, Motive provides a convenience function, `interp`, which takes a series of values representing notional accelerations over time and builds a corresponding interpolation function. This is done by calculating velocities from accelerations and then distances from velocities by integration.

The distances are normalized and a continuous interpolation function is produced by cubic spline interpolation of the normalized distances. For example, `(interp [10 0 0 -5 -5])` defines an interpolation function that accelerates by 10 units and then maintains speed before gently slowing down. The acceleration values are normalized, the values of 10 and -5 have no meaning outside their relative value. As described in Sec. 2.2.2, interpolation functions have a domain $[0, 1]$ but simple movements occur for a time duration, t; Motive manages the mapping between the time domain of the movement, defined as $[0, t]$ and the $[0, 1]$ domain of the interpolation function.

Using the previously introduced `quat` and `interp` functions a simple movement may be defined. Recall from Sec. 2.2.2 that simple movements also include a time duration attribute. A simple movement that rotates $45\,^\circ$ about the x-axis with an acceleration at the start and deceleration at the end, and a total time of 3.5 seconds is written using the `simple` function as seen in Listing 3.1.

```
(simple (quat 0.7854 (axis -1 0 0))
        (interp [10 0 0 -5 -5])
        3.5)
```

Listing 3.1: *A simple movement.*

As stated in Definition 2.5, super movements are mappings of joints to sequences of simple movements. Also from Definition 2.5, sequences of simple movements are a collection of real number and simple movement pairs. The real number value is the length of time, in seconds, that the paired simple movement should be delayed. Listing 3.2 is an example of a super movement that will apply two simple movements to the right shoulder joint. When the super movement begins, the first simple movement will be applied immediately since the time delay is 0.0. Once the first simple movement has completed, the second simple movement will be applied after a 1.0 second delay.

Motive currently provides implementations for some of the transformation functions described in Sec. 2.2.4. For example, the `combine-interp` function is an implementation of Definition 2.7. The implementation in Listing 3.3 accepts the weight factor as an explicit

```
(super {right-shoulder [[0.0 (simple (quat 0.7854 (axis -1 0 0))
                                     (interp [10 0 0 -5 -5])
                                     3.5)]
                        [1.0 (simple (quat 0.7854 (axis -1 0 0))
                                     (interp [5 5 5 0 -15])
                                     2.0)]]})
```

Listing 3.2: *A super movement.*

argument. An anonymous function,

$$f : [0, 1] \rightarrow [0, 1], x \mapsto \alpha interp_1(x) + (1 - \alpha) interp_2(x)$$

, is returned.

```
(defn combine-interp
  "InterpFn x InterpFn x Real -> InterpFn"
  [i1 i2 alpha]
  (fn [x] (+ (* alpha (i1 x))
             (* (- 1 alpha) (i2 x)))))
```

Listing 3.3: *Combine interpolation functions.*

Functions that close a movement definition by supplying concrete values to placeholders are also possible. Listing 3.4 is essentially a map from a set containing one item, an open simple movement, to a set of closed movements that differ only by the axis of rotation.

```
(defn close-movement-axis
  "Axis -> SimpleMovement"
  [axis]
  (simple (quat 0.5 axis) (interp [100 0 100 0]) 3))
```

Listing 3.4: *Close a movement definition with the given axis.*

3.2 Visualization

Motive provides a three-dimensional view of a body model and the application of movements. The rendering engine used is jME, [8], a scene graph based graphics library for Java.

The body model was built programmatically from mesh cylinders and scene graph nodes. The node hierarchy for the body begins at the hip, extends up the trunk to the neck from which the shoulders and head are connected. Each arm has two nodes that represent the shoulder and elbow joints and two cylinder meshes that represent the upper and lower arm segments.

Motive's visualization capability was written in Clojure using its Java interoperability to extend Java classes. In order to integrate Clojure with jME, it was originally necessary to subclass a jME class. At that time, Clojure was capable of extending Java classes, but with limitations that prevented it from being used in this scenario. To overcome this barrier, the author submitted several patches that improved Java static field resolution and enabled access to superclass implementations of overridden methods. Those patches are now a part of the Clojure language implementation.

Definitions, as described in Sec. 3.1, entered at Motive's interactive prompt can be used to animate the body model in real time. For every rendered frame of the scene, the orientation of all joints is calculated and the scene is updated. Listing 3.5 shows the Motive `move` function applying a super movement to the body model. This particular movement is a salute that brings the right arm from a neutral position to the chest. Fig. 3.1 contains several frames from the animation sequence of the salute movement being applied to the body model.

```
(move {right-elbow [[0 (simple (quat 1.8 (axis -1 0 0))
                               (interp [1000 0 10000 0 0 -4000])
                               6)]]
       right-shoulder [[3 (simple (quat 1.7 (axis -0.5 1 0))
                                  (interp [50000 -1000 -50000])
                                  4)]]})
```

Listing 3.5: *A salute.*

The salute movement involves two simple movements, one on the right elbow and the other on the right shoulder. The right elbow movement begins first, and brings the forearm towards the shoulder. The right shoulder movement begins when the elbow movement is

halfway completed and brings the forearm towards the chest and into a salute pose.

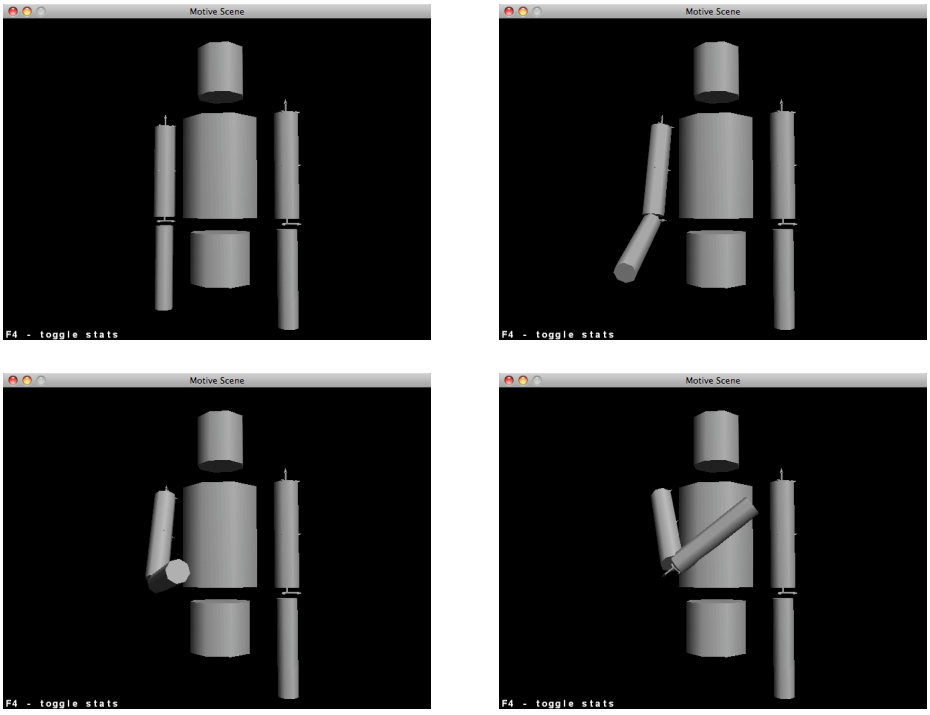

Figure 3.1: *Captured frames of salute as defined in Listing 3.5.*

Listing 3.6 shows the definition of a flagging wave movement being applied to the body model with the `move` command. Frames of the movement being applied to the body model are shown in Fig. 3.2. This movement involves four joints: the left and right shoulders and the left and right elbows. The arms are raised up above the head and then shoulder rotations wave the arms back and forth. The complete movement involves three joint rotations of both elbows and seven joint rotations of both shoulder joints. Five of the shoulder movements are used to wave the raised arm back and forth and involve repeated movements. It is possible to literally define the repeated movements multiple times, but instead Clojure's built-in support for cycling over a finite sequence (`cycle`) is used.

```
(move {right-elbow [[0 (simple (quat 2.35 (axis -1 0.25 0))
                               (interp [1 0 0])
                               2.5)]
                    [0 (simple (quat 2 (axis 1 -0.25 0))
                               (interp [1 0 0])
                               2)]
                    [0 (simple (quat 0.3 (axis -1 0 0))
                               (interp [1 0 0])
                               0.75)]]
       right-shoulder [[1.5 (simple (quat 2.45 (axis -1 0.2 0))
                                    (interp [1 0 0])
                                    3)]
                       [0 (simple (quat 0.45 (axis 0 0 1))
                                  (interp [1 0 0])
                                  0.35)]
                       (take 5 (cycle [[0 (simple (quat 0.9 (axis 0 0 -1))
                                                  (interp [1 0 0])
                                                  0.75)]
                                       [0 (simple (quat 0.9 (axis 0 0 1))
                                                  (interp [1 0 0])
                                                  0.75)]]))]
       left-elbow [[0 (simple (quat 2.35 (axis -1 -0.25 0))
                              (interp [1 0 0])
                              2.5)]
                   [0 (simple (quat 2 (axis 1 0.25 0))
                              (interp [1 0 0])
                              2)]
                   [0 (simple (quat 0.3 (axis -1 0 0))
                              (interp [1 0 0])
                              0.75)]]
       left-shoulder [[1.5 (simple (quat 2.3 (axis -1 -0.2 0))
                                   (interp [1 0 0])
                                   3)]
                      [0 (simple (quat 0.45 (axis 0 0 -1))
                                 (interp [1 0 0])
                                 0.35)]
                      (take 5 (cycle [[0 (simple (quat 0.9 (axis 0 0 1))
                                                 (interp [1 0 0])
                                                 0.75)]
                                      [0 (simple (quat 0.9 (axis 0 0 -1))
                                                 (interp [1 0 0])
                                                 0.75)]]))]})
```

Listing 3.6: *A flagging wave.*

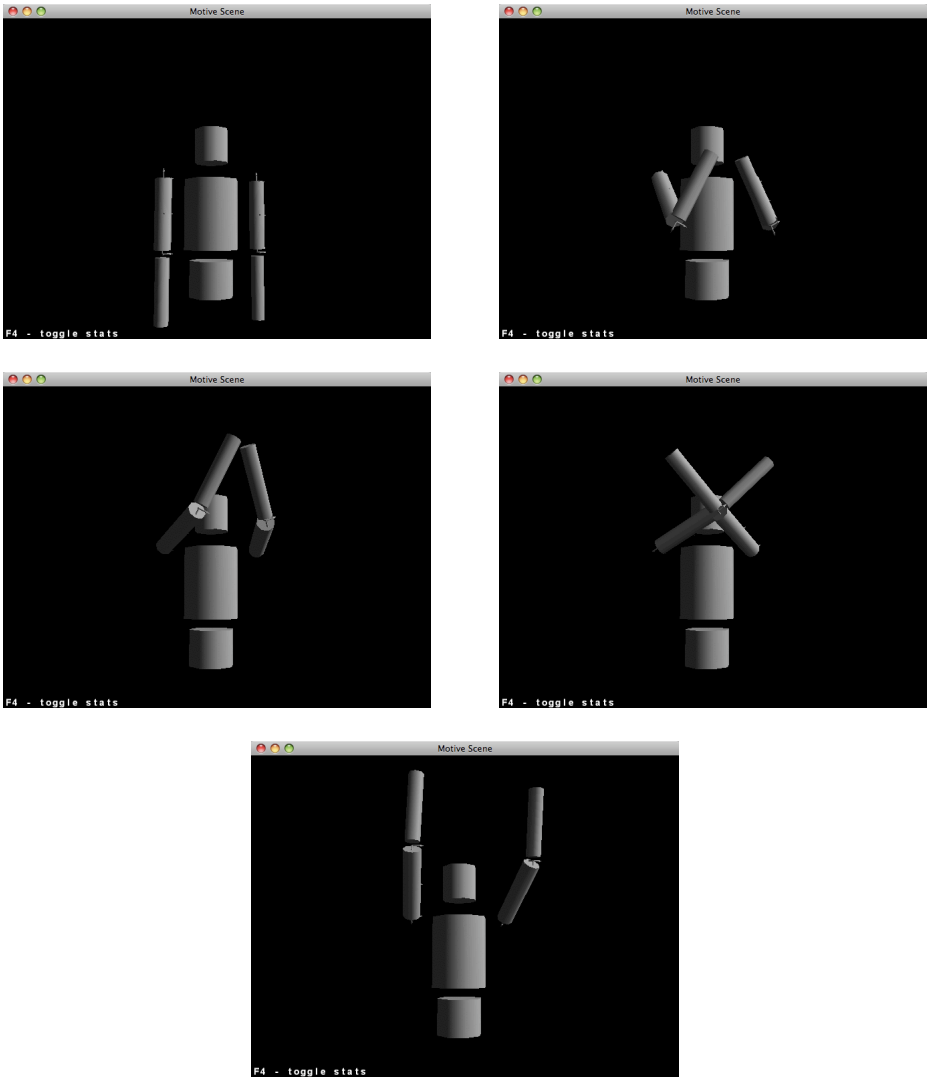

Figure 3.2: *Captured frames of flagging wave as defined in Listing 3.6.*

Chapter 4: Discussion

The symbolic representation developed in this thesis differs from others by its ability to scale in generality. This scalability is provided by support for sets of possible values as parameters. Even with a fixed limb model, a single movement definition is able to represent an infinite set of movements as easily as a single instance.

A system for defining and operating on joint movement bridges the theoretical world of simulation and computation with the realities of human movement. Once transformed to their system representation, captured movements may be manipulated to account for physical limitations of an individual's neuromusculoskeletal system and support human rehabilitation efforts.

The current system is useful but requires further development; it is incapable of representing conical movement. This is due to the assumption that spherical linear interpolation should be done between the orientation of the joint at the start of the movement application and the target orientation that is calculated by multiplying the start quaternion by the movement rotation. In order to represent conical movements additional information is required.

As presently implemented, all defined joint movements must be rotation about a fixed axis. Imagine a joint as a sphere in the center of a larger sphere where the larger sphere's radius matches the length of the limb segment attached to the joint. A joint movement from one pose to the next will be a rotation tracing the shortest path over the surface of the sphere. No curved paths are properly definable without additional information. Fig. 4.1 depicts two different paths between the same pair of poses. Extending the representation model to express movements of this kind will be a natural progression of the system.

An initial goal of the thesis was to create a representation of movement for building a library of human movement. The library should contain general definitions of entire

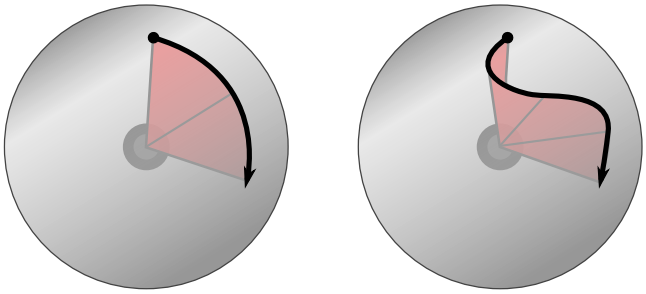

Figure 4.1: *Joint rotations traced on the surface of a sphere. The start and end orientations are identical, but the path of movement differs.*

movement classes and corresponding class exemplars. General definitions are declared by supplying a definition parameter that is a set of possible values rather than a specific value.

The class of the raise hand movement may be described as: *a*) shoulder orientation about x-axis with an angle in the range of $[\frac{\pi}{4}, \pi]$, *b*) elbow orientation about x, y, and z axes are within a small threshold of π, 0, and 0, respectively, and *c*) any speed or time duration . Notice that the movement class description refers to joint *orientations*, not rotations; movements are only members of this class when they move the limb into a state meeting the class criteria. In this aspirational database of movement, a user may provide an initial body state and movement definition and a list of all matching classes will be returned.

Assuming a movement has been classified as raise hand, analysis of the movement attributes may enable further classification; if the movement was performed slowly, then the classification becomes raise hand slowly. Additional knowledge, e.g. the subject is a student sitting in a classroom, is enough context that the emotional state of the subject, say nervous, may be inferred. To support machine classification of movements while reducing supervision, a method for identifying movements composed of simpler movements is needed.

A movement grammar and associated parser can be used in conjunction with movement classification to identify movement sequences as primitives in a higher-order movement. Returning to the raise hand class, any of: a shoulder movement, an elbow movement, or

a combination of shoulder and elbow movements; may be classified as raise hand. This relationship can be captured in a production rule as demonstrated in Fig. 4.2.

$$\begin{aligned} \underline{\text{raise hand}} &\rightarrow \underline{\text{shoulder movement}} \\ &\rightarrow \underline{\text{elbow movement}} \\ &\rightarrow \{\underline{\text{shoulder movement}}, \underline{\text{elbow movement}}\} \end{aligned}$$

Figure 4.2: *Production rule for raise hand*

Chapter 5: Conclusion

The goals of defining and implementing a symbolic system for the representation of movement and implementing an integrated environment for user interaction and movement visualization were met. The current syntax for interactively defining movement is sufficient in expressivity, but a higher-level syntax could be added to improve the user experience.

The most glaring deficiency of the current system is its limitation of only representing movements that involve planar rotations. The result being that conical movements, which require a non-planar joint rotation, are unrepresentable. There has been research on non-planar rotations and quaternions [14] that may provide an elegant solution to the problem but further investigation is required.

As the movement representation implemented in this thesis is capable of describing precise movements, importing movement captured by motion tracking devices into the symbolic representation should be straightforward. Once imported, the captured movement would be equivalent to literally defined movement and would be used in the same manner.

The purpose of building this representation of movement is to provide a foundation for further analyses and research of movement. Building a library of classified movements defined by a grammar could provide the basis for a probabilistic movement classifier.

There is also further work with regard to body models to pursue. Adding constraints to the body model would lead to more realistic simulation of the human body. The constrained model could be used to simulate physical disabilities and identify compensatory movements used for performing various tasks affected by a disability. Movement retargeting, which involves transforming a movement defined for a particular body model to a different body model, is being actively researched. The function of retargeting a movement to a different body model may be possible to incorporate into this movement system.

Bibliography

Bibliography

[1] Calais-Germain, B. *Anatomy of Movement*. Eastland Press, Seattle, USA, 1993.

[2] Calvert, T. W., and Chapman, J. Notation of movement with computer assistance. In *ACM '78: Proceedings of the 1978 annual conference* (New York, 1978), pp. 731–736.

[3] Eshkol, N. *The quest for T'ai Chi Chuan*. Tel Aviv University, Tel Aviv, Israel, 1988.

[4] Eshkol, N., and Wachmann, A. *Movement Notation*. Weidenfeld & Nicolson, London, 1958.

[5] Franklin, W. R. Efficient iterated rotation of an object. *IEEE Trans. on Computers 32*, 11 (1983), 1064–1067.

[6] Hudak, P., and Fasel, J. H. A gentle introduction to haskell. *SIGPLAN Not. 27*, 5 (1992), 1–52.

[7] Isenberg, D. R., and Kakad, Y. P. Computed torque control of a quaternion based space robot. In *ICSENG '08: Proc. of the 2008 19th International Conference on Systems Engineering* (Washington, DC, 2008), IEEE Computer Society, pp. 59–64.

[8] jMonkeyEngine. jmonkeyengine.com. `http://www.jmonkeyengine.com`, April 2009.

[9] Moran, G., Fentress, J. C., and Golani, I. A description of relational patterns of movement during ritualized fighting in wolves. *Animal Behavior*, 29 (1981), 1146–1165.

[10] Murray, R. M., Sastry, S. S., and Zexiang, L. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA, 1994.

[11] Nakanishi, J., Cory, R., Mistry, M., Peters, J., and Schaal, S. Operational space control: A theoretical and empirical comparison. *Int. J. Rob. Res. 27*, 6 (2008), 737–757.

[12] Perez, M. A. *Dual quaternion synthesis of constrained robotic systems*. PhD thesis, 2003.

[13] Sandewall, E. Programming in an interactive environment: the "lisp" experience. *ACM Comput. Surv. 10*, 1 (1978), 35–71.

[14] Shoemake, K. Animating rotation with quaternion curves. In *SIGGRAPH '85: Proc. of the 12th annual conference on Computer graphics and interactive techniques* (New York, 1985), pp. 245–254.

[15] Singh, B., Beatty, J. C., and Ryman, R. A graphics editor for benesh movement notation. In *SIGGRAPH '83: Proc. of the 10th annual conference on Computer graphics and interactive techniques* (1983), pp. 51–62.

[16] The Internet Encyclopedia of Science. Skeletal muscles and muscle groups. http://www.daviddarling.info/encyclopedia/S/skeletal_muscle_groups.html.

[17] Tratt, L. Domain specific language implementation via compile-time meta-programming. *ACM Trans. Program. Lang. Syst. 30*, 6 (2008), 1–40.

[18] van Waveren, J. M. P. From quaternion to matrix and back, February 2005. http://cache-www.intel.com/cd/00/00/29/37/293748_293748.pdf.

[19] Wilke, L., Calvert, T., Ryman, R., and Fox, I. From dance notation to human animation: The labandancer project: Motion capture and retrieval. *Comput. Animat. Virtual Worlds 16*, 3-4 (2005), 201–211.

[20] Yu, T., Shen, X., Li, Q., and Geng, W. Motion retrieval based on movement notation language: Motion capture and retrieval. *Comput. Animat. Virtual Worlds 16*, 3-4 (2005), 273–282.

Curriculum Vitae

Matthew Revelle received his Bachelor of Science in Computer Science from the University of Mary Washington in 2003. He worked as a Software Engineer for several years before applying to the Masters of Science in Computer Science program at George Mason University. While progressing through the Master of Science in Computer Science program at George Mason University he took a brief hiatus to serve as a technical lead and founding member of a software startup. He returned to the program and to a position as a Software Engineer.