A General Joint Component Framework For Realistic Articulation in Human Characters

Wei Shao*
New York University

Victor Ng-Thow-Hing†
Honda Research Institute USA, Inc.

Abstract

We present a general joint component framework model that is capable of exhibiting complex behavior of joints in articulated figures. The joints are capable of handling non-orthogonal, non-intersecting axes of rotation and changing joint centers that are often found in the kinematics of real anatomical joints. The adjustment of joint articulation is done with a relatively small set of intuitive parameters compared to the number of articulations in the motions they parameterize. This is done by making various linear and nonlinear joint dependencies implicit within our framework. An animator is restricted from putting the skeleton in an infeasible pose. We have used our joint framework model to successfully model highly-articulated complex joints such as the human spine and shoulder.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modelling—Modelling packages; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction technique;

Keywords: biomechanical joint, human modelling, articulated figure, joint modelling

1 Introduction

The need to animate realistic depictions of human characters is an important element contributing to the appeal and success of interactive graphics applications. 3-D humanoids are often the physical embodiment of the interactive interface, either serving as an avatar for the player, or as representations of other autonomous characters in the virtual environment. While recent research has focused on better geometric representations [DeRose et al. 1998], anatomy-based modelling [Scheepers et al. 1997; Wilhelms and Van Gelder 1997], and pose-based skin deformation [Lewis et al. 2000; Sloan et al. 2001; Wang and Phillips 2002] for representing and animating the exterior skin and muscle deformations of humans, the underlying articulated body representation that drives these methods has generally been unchanged since its first use in computer animation in the 1980's[Tost and Pueyo 1988].

Introducing accurate biomechanical joint models to the traditional hierarchy of joint transformations can lead to improved realism in human character animation. Interactive applications such as 3-D computer games and virtual reality have benefited from improvements in graphics hardware by depicting humanoid characters with greater levels of detail in their geometric surface models. However, studies in perception of human motion with different geometric models have suggested that observers may be more sensitive to motion changes if polygonal models are used compared to stick figures [Hodgins et al. 1998]. As the majority of 3-D interactive applications use polygonal models with increasing detail, observers may be more sensitive to noticing unrealistic joint motions in animations, such as in the shoulder and torso regions of the body. This is especially relevant to sport simulations where perceived athleticism is tied to the coordination of movement in virtual human players. For animation techniques that deform an outer skin model based on skeleton motion, accurate joint transformations of bone segments can lead to better performance of these methods by improving the association between skin movement and the underlying bones in the case of pose-based deformations and modelling more accurate muscle deformations in response to skeletal movement in the case of anatomy-based models.

Despite the continued work to develop better joint models in both the biomechanics and computer graphics communities (see Section 2), their adoption in mainstream computer graphics such as 3-D games has been limited. Part of the reason may be the misperception that complex joint models will require more user handles for an animator to control or that these models require large changes in software infrastructure that make their use incompatible with popular 3-D modelling packages, such as *Maya*[Alias/Wavefront 1999]. Indeed, the majority of commercial software allows skeleton hierarchies to be built only as a tree of ball-and-socket joints between bone segments. The data structures for bone segments consist of a transformation matrix that is relative to its parent coordinate frame in the tree. The motion of a single joint is usually restricted to the relative motion between two adjacent bone segments.

This paper introduces a general joint component model framework that allows the modelling of joint expressions over several bone segments, biomechanically accurate joints with non-orthogonal rotation axes, changing joint centers, closed loops, and intuitive parameters to configure these joints. We utilize several previous concepts developed for accurate joint modelling, and allow them to be expressed together within a common framework that hides the complexity from an animator. Each joint component is a mapping that takes a set of inputs to produce outputs that can be attached to other components in the framework to produce a network whose final outputs correspond to the transformations of individual bone segments. The inputs of the network correspond to the set of parameters that the animator or an animation algorithm will modify to control articulation of the human model (Figure 1). In this manner, a relatively few set of intuitive parameters can be designed to effectively control complex articulations in the human body. We demonstrate the versatility of the model by building biomechanically detailed shoulder and spine examples which can be manipulated in real-time.

Previous developments in joint modelling are discussed in Section 2, highlighting ideas we adopt in our component model frame-

*e-mail: weishao@cs.nyu.edu
†e-mail:vng@honda-ri.com

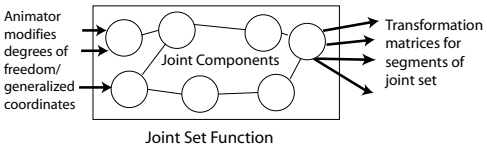

Figure 1: Overview of the joint component framework

work. Section 3 gives an overview of the general structure of our joint component framework, followed by a more detailed description of our joint components in Section 4. In Section 5, we illustrate how the framework can be used to model two relatively complex joints (the spine and shoulder). Section 6 concludes with discussion and future work.

2 Previous Work

The origins of articulated joint models for human character representations can be found in the study of kinematics of robotic manipulators [Craig 1989]. Early animation systems, such as PODA [Girard and Maciejewski 1985] made use of the Denavit-Hartenberg link parameter notation from robotics to represent figures with articulated limbs. Although the notation is a convenient way to relate coordinate frames between adjacent segments with four parameters, each parameter set only describes a single degree of freedom between two segments. Multiple sets of parameters must be combined to achieve multiple degree of freedom (DOF) joints. For complex articulations, a higher level of organization to provide a convenient, unified interface for adjusting multiple joint DOFs to an animator is desirable.

Although Euler angles[Craig 1989] are often used to express segment orientations, quaternions [Shoemake 1985] and exponential maps [Grassia 1998] have emerged as excellent alternatives that have desirable interpolation properties as well as avoiding singularities inherent with Euler angles. Nevertheless, Euler angles have persisted in popularity probably because their degrees of freedom have natural analogs to motions descriptions such as twist, flexion-extension and abduction-adduction in human movement. Our framework does not restrict the choice of parameterization for orientation, but allows the choice of best representation for the situation.

In the area of biomechanics, physiological joints have been shown to have many complexities that are often neglected in graphical models. For example, biomechanists routinely specify joints with several non-orthogonal, arbitrary axes of rotation [Delp and Loan 1995] that are better aligned to bone articulation. Many joints have translational components and changing centers of rotation, including the knee which is traditionally simplified as a single DOF hinge joint [Bull and Amis 1998]. In joints like the shoulder, the closed loop consisting of the clavicle, scapula and thoracic surface of the rib cage creates a coupling between the articulations of all these joints. Several groups model this situation by enforcing a constraint on the scapula to stay on the surface of an ellipsoid approximating the rib cage [Garner and Pandy 1999; Maurel and Thalmann 2000]. Other structures, like the human spine, exhibit a high degree of coupling behavior between the vertebrae. Monheit and Badler [Monheit and Badler 1991] exploit this fact to develop a kinematic model of the human spine that exhibits flexion/extension, lateral bending and axial twist rotation. Our framework is designed to accommodate all these desirable biomechanical characteristics.

Finally, there have been several animation systems that focus on modelling accurate, anatomical joints with similar features to our framework. Maciel et al. [Maciel et al. 2002] develop a model that incorporates joints that can translate and rotate together on a plane and have joint limits that dynamically change with the DOF of any joint. Each DOF is associated with an axis of rotation or translation for a single segment. In contrast, we have expanded our definition to allow a DOF to control a coordinated set of motions over several segments. Joint sinus cones [Maurel and Thalmann 2000; Wilhelms and Van Gelder 2001] have been introduced to the graphics community to provide a better mechanism of restricting joint angle ranges for ball-and-socket joints. We incorporated joint cones into our framework to produce natural limits on limb motion as DOFs are adjusted. The Peabody system [Badler et al. 1992] collects joints into joint groups that have group angles to configure the joint group's segments. This system is closest to our philosophy of providing a higher level organization to coordinate individual joints. Although our framework allows more generality in the joint models by allowing changing joint centers, surface constraints and joint sinus cones for joint limits on ball-and-socket joints, we believe the Peabody system could similarly be extended to accommodate these features.

The H-Anim specification [Humanoid Animation Working Group 1999] describes a standardized humanoid joint hierarchy for the purpose of avatar representation. Custom joints can be added only if they do not interfere with the movement of standard joints. Although a standard human representation is important for avatar exchange and compatibility in different software, the flexibility to define new coordinated articulations is hampered by constraints enforced by the hierarchy.

Judging from the extensive array of previous joint models, we can see that no single representation is best for capturing all joints. In fact, specialized joint models are often needed, as in the shoulder and spine. Similarly, the complexity of these articulations should be made accessible to animators by providing intuitive controls. In applications where physiological consistency is desired, an animator should not be allowed to configure a skeleton into a non-natural, infeasible posture. These guidelines influenced the design of our general joint component model.

3 General Joint Component Model

Some terminology and notation is introduced to clarify the discussion that follows. An articulated figure consists of a set of *segments* s that can express only rigid body motion. The segments are related to each other by a hierarchy where the motion of segment s is expressed relative to its parent p in the form of a 4x4 transformation matrix ${}^{p}_{s}T$ (we will remove the superscript p when we do not need to reference the parent segment). A segment can have no parent, ${}^{0}_{s}T$, implying that its motion is relative to the world coordinate frame. Therefore, the segments of a single articulated figure can be partitioned into several hierarchical trees. This is useful if an articulated figure contains free-floating segments.

A *joint set* J contains one or more segments whose configuration is described by independent *degrees of freedom (DOF)* or *gen-*

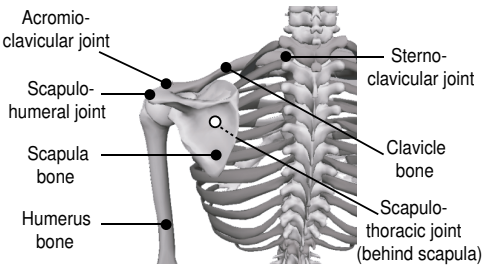

Figure 2: Shoulder joint set

eralized coordinates (GC), q_J. For each segment in the joint set, its relative motion with its parent is described as an articulation or *joint*. For example, the shoulder joint set consists of four bone segments (clavicle, scapula, thorax and humerus) with four articulations (sterno-clavicular, acromio-clavicular, scapulo-humeral and scapulo-thoracic joints) as shown in Figure 2. Segments do not have to be adjacent to each other within a joint set, implying a segment's parent can reside in a different joint set.

Mathematically, the transformation matrix for each segment's joint in J is expressed as a changing transformation matrix function, ${}_sT(q_J)$. For each joint set J, we define a *joint set function* as a mapping of its generalized coordinates to the transformation matrix functions for each segment (Figure 1):

$$f_J : \{q_J\} \rightarrow \{{}_sT | s \in J\}.$$

By partitioning the motion of segments of an articulated figure into several joint sets, it is possible to kinematically control complex articulations with only a few DOFs. In Section 5, we implemented a 24 vertebrae human spine using three joint sets for the cervical, thoracic and lumbar regions, with each region having three DOFs for flexion/extension, side-bending and axial twisting.

To implement joint set functions, a set of building blocks called *joint components* were designed. Each joint component implements a cohesive function, facilitating its reuse in different contexts. A joint component is formally defined as a mapping,

$$j_\Lambda : \Theta \rightarrow \Omega,$$

where Λ is a list of parameters that configure a joint component to describe specific features such as joint limits and axes of rotation. The sets, Θ and Ω can have scalar, vector or matrix elements and correspond to the inputs and outputs of the joint component, j_Λ respectively. A joint set function comprises a network of joint components that is created by connecting the output of one component to the inputs of one or more other components. Generalized coordinates feed into the network with transformation matrices for segments produced as output. A relatively small number of simple joint components can be combined to create a diverse array of behaviors. This framework allows new types of joint components to be added and used with existing components with minimal coupling between modules.

4 Joint Components

In our current framework, we have implemented a repertoire of seven types of joint components: matrix multiplication, one-to-many mapping, compensation, rotation, dependency, joint cone, and scapula constraint. Each of these will be discussed in the following subsections.

4.1 Matrix Multiplication Component

The *matrix multiplication component* takes as input a list of several matrices and multiplies them together to produce a single transformation matrix as output. The order of elements in the list determines the multiplication order. The output can either be the final transformation that will be applied to the corresponding joint or an intermediate result that will be used as input to other components. This component is very useful as many transformations can be expressed as matrix decompositions with intuitive parameters for an animator [Shoemake and Duff 1992].

4.2 One-to-Many Mapping Component

A *one-to-many mapping component* has a single scalar input and produces a vector of one or more joint variables, which can be interpreted as angles or translational units for revolute or translational joints. This mechanism allows a single scalar DOF to control the articulations of more than one joint. For example, we have implemented a knee model where a single generalized coordinate is the common parameter of several cubic spline functions that evaluate the Euler angle rotations and translations for the patella, fibula and tibia bones (Plate 1).

Within this component, each element of the output vector can have its own linear or nonlinear function in its connection from the input scalar. These functions can implement different rates of change for several joint variables, unit conversions, and joint limits for one or more joints. For example, the domain $[0, 1]$ can be mapped to the limits $[\theta_{min}, \theta_{max}]$ for a rotation angle θ.

4.3 Compensation Component

In a standard hierarchical skeleton tree, the transformations of a parent segment are inherited by the child segment. However, this may produce undesirable behavior in some situations. For instance, if we wanted to shrug the shoulders of a human model, the rotation in the clavicle would propagate to the humerus, causing the humerus to rotate away from the body (Figure 3). An animator may want to keep the orientations of the humerus and clavicle independent. The Peabody system handles this by applying a corrective angle rotation that compensates for the parent's propagated rotation [Badler et al. 1992]. The *compensation component* generalizes this concept to cancel out the orientation transformation (created by any rotation parameterization) of any ancestor segment (not just the direct parent) for a particular segment while maintaining connectivity at the joint with its segment by adjusting the translation of the segment. This allows a segment to have orientation with respect to the world frame.

The compensation component for a particular segment s takes as input a single transformation matrix of an ancestor segment and produces a matrix which cancels out the undesired orientation change caused by the rotation of that ancestor. Depending on how far up the skeleton hierarchy tree we wish to cancel out orientations, a segment can have a compensation component for each ancestor. The outputs of each compensation component are then multiplied using a *matrix multiplication component* to produce a matrix that compensates for all the undesired movements caused by a segment's ancestors up to a certain level in the skeleton tree. In the shoulder joint set, two compensation components are used to create an independent humerus orientation from the scapula and clavicle rotation. The first component cancels the effects of the acromio-clavicular joint (connecting the scapula to the clavicle), and the second component nullifies the sterno-clavicular joint (connecting the clavicle to the sternum of the rib cage).

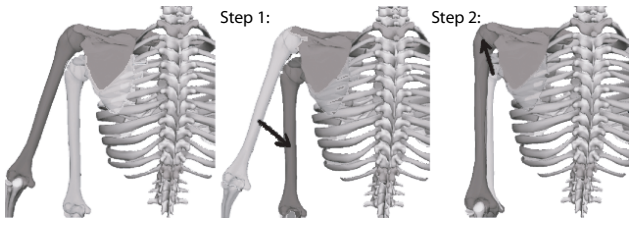

Figure 3: The two steps for computing the compensation matrix

The computation of the compensation matrix is done in two steps. First, the inverse of the ancestor segment's transformation matrix is computed to cancel out its movement. At this stage, the segment has moved back to its original position and orientation before any of the ancestor transformations have been applied. The segment now has to be reconnected to the same coincident joint location it shared with its parent. In the second step, a corrective

translation is calculated as the displacement of the segment's local origin caused by the ancestor's transformation matrix.

In Figure 3, the scapula, which is the parent of the humerus, is itself a child of the clavicle. On the left, the clavicle rotates from its reference position (light gray), to a new configuration (dark gray). The first step of applying the inverse transformations of the scapula and clavicle to the humerus results in the configuration shown in the center image. The humerus is now disjointed from the scapula. In the second step, we apply the corrective translation (shown by the black arrow in the right image) to the humerus to reunite it with the scapula.

4.4 Rotation Component

The *rotation component* is designed to accommodate non-orthogonal, arbitrary axes of rotation with changing joint centers that are a function of generalized coordinate values. Joints with multiple DOF rotations are created by combining rotation components, each of which produces a rotation matrix for a single axis rotation. An important simplifying assumption being made is that the joint centers for each axis rotation is independent of the rotations about the other axes. While this may not be the case in reality, reasonable articulations were observed in the joints we created.

For each rotation component, the following rotation parameters can be provided:

1. A list of n consecutive angle intervals: $[a_0,a_1), [a_1,a_2), \ldots, [a_{n-1},a_n]$
2. For each angle interval $[a_{i-1},a_i)$, a rotation center point $c_i =< c_x,c_y,c_z >$
3. a common rotation axis x.

Each angle interval has a different joint rotation center. The ability to model a changing joint rotation center is important to accurately describe rotations in the knee and humerus of the shoulder [Kapandji 1982b]. There may be only a single angle interval (and corresponding joint center) defined for a rotation component. With these rotation parameters and an input rotation angle $\alpha \in [a_0,a_n]$ (which can be derived from the output of other joint components), the final output rotation matrix R for the joint component is computed as follows:

$$\begin{aligned} R &= M_n \times M_{n-1} \times \ldots \times M_1 \\ M_i &= T_i^{-1} \times Q_i \times T_i \\ T_i &= \begin{pmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ T_i^{-1} &= \begin{pmatrix} 1 & 0 & 0 & c_x \\ 0 & 1 & 0 & c_y \\ 0 & 0 & 1 & c_z \\ 0 & 0 & 0 & 1 \end{pmatrix}, \end{aligned} \tag{1}$$

where R, M_i, T_i, T_i^{-1}, and Q_i are all transformation matrices. Q_i is computed from a quaternion representing the rotation of angle β_i around axis x. The angle β_i is determined by

$$\beta_i = \begin{cases} a_i - a_{i-1} & \text{if } \alpha > a_i \\ \alpha - a_{i-1} & \text{if } a_{i-1} < \alpha < a_i \\ 0 & \text{otherwise} \end{cases}.$$

The final rotation of the angle α is the cumulative result of a sequence of quaternion rotations of smaller angles β_i. Each of these quaternion rotations has its own center represented in the T_i and T_i^{-1} translation matrices. This implements a changing joint rotation center of a joint. An example is given in Figure 4.

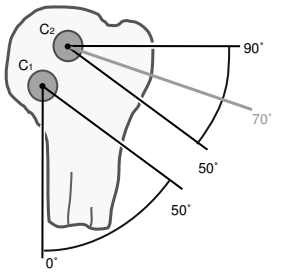

Figure 4: Floating center of shoulder abduction. Abduction of upper arm takes c_1 as rotation center during 0 to 50 degrees and takes c_2 during 50 to 90 degrees rotation. For an abduction angle of 70 degrees, first a 50 degree rotation is carried out around c_1, and the remaining 20 degrees uses c_2 as the center of rotation.

Notice that for a given α that falls into a certain interval $[a_{j-1},a_j)$, all M_k $(0 < k < j)$ will not depend on α since each of their β_k is a constant equal to $a_k - a_{k-1}$. We can precompute and store all the matrices M_k for the full angle interval rotation to reduce computation. Furthermore, the cumulative matrix product R in Equation 1 for the j^{th} interval can have $M_{j-1} \times M_{j-2} \times \ldots \times M_1$ precomputed and retrieved to update the articulated figure motion at interactive rates.

4.5 Dependency Component

The *dependency component* allows the modelling of coupling behavior between different joints within a joint set. In each dependency component, a pair of joints are specified, one as the active joint a that drives the other passive joint p. The movement of a DOF of p is set to be dependent on a DOF of a through a mapping function. The actual nature of the mapping function used in the dependency component can be any linear or nonlinear function. Interpolating splines are often convenient to match dependency relations to experimental data samples. Typically, the DOF corresponds to Euler angles that define the rotation matrix of each joint. The dependency component takes two input Euler angles, one from each joint, and contains a mapping function to output a modified angle for the passive joint. We implemented several types of mapping relationships:

1. One-to-one mapping: For any given DOF value of a, a DOF value for p is defined. For instance, the rotation of the scapula around an axis perpendicular to its outward surface tangent plane is almost linearly dependent on the abduction of upper arm. A linear one-to-one mapping can capture this relationship.
2. One-sided bound mapping: This is a one-to-one map where values of p are bounded on one side by a lower or upper limit that is a function of a DOF of a. An example of this is the dependency between the abduction of the humerus bone and the elevation of the clavicle bone. The higher the upper arm is raised, the more restricted is the vertical movement of the shoulder's clavicle. The restriction is due to a lower limit placed on clavicle elevation, which can be implemented as a one-sided bound that is dependent on the amount of abduction of the humerus.
3. Two-sided bound mapping: The value of a DOF of p is bounded on both sides by limits dependent on a DOF of a.

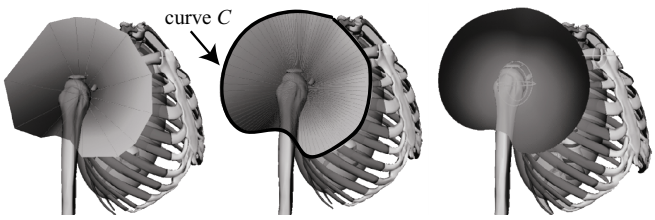

Figure 5: Joint limit cone. On the left, a list of 14 points is used to define the curve C. In the middle, the cone is refined using subdivision. On the right, the cone is shaded based on the twist limits on each point. Darker shades indicate a more restricted limit range.

Again using the shoulder as an example, when the left upper arm is rotating in the horizontal plane from the left side to the front right of the body, the horizontal movement of the shoulder (at the clavicle bone) becomes more restricted. A similar phenomenon occurs when the left upper arm is rotating to the back of the body. Since there are both upper and lower limits, a two-sided mapping is appropriate.

4.6 Joint Cone Component

Joint sinus cones [Maurel and Thalmann 2000; Wilhelms and Van Gelder 2001] have been used to provide a better mechanism for joint limits for ball-and-socket joints than pairs of Euler angle bounds for each joint DOF. We have extended the joint cones in [Wilhelms and Van Gelder 2001] to accommodate changing joint centers that can occur with our rotation components.

In a *joint cone component*, the joint sinus cone is defined using a reference point p and a space curve C. The reference point p is the apex of the cone and is located at the joint center. The curve c creates a boundary of the bottom of the cone and is defined by an initial list of user-selected control points. Subdivision rules are used to refine and smooth the curve. An additional vector v_{rest} is defined and positioned at p so that it lies in the same direction as the bone's longitudinal axis at its rest configuration. This cone provides a way of bounding the movements of two DOF of a joint, such as abduction/adduction and flexion/extension in the humerus at the shoulder. To limit the third twist DOF, an additional pair of angle bounds is associated with each control point on the curve c and the point v_{rest}. During the refinement process, new interpolated bound pairs are computed for the new control points produced by subdivision. To get a pair of twist limits for any given configuration within the cone, interpolation is performed at run-time (Figure 5).

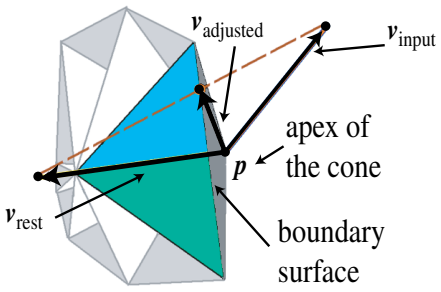

Figure 6: The projected configuration

Joint cone components are used to check for legal joint configurations and project illegal orientations back to the boundary curve. A joint cone component is always attached to one specific joint, say joint j. It takes as input the transformation matrix T_{input} of j, which is produced by rotation components. T_{input} transforms the vector v_{rest} to v_{input} to test if the bone's orientation is within the joint cone. If it is, T_{input} is passed out of the joint cone component. Otherwise, a new transformation matrix $T_{adjusted}$ is computed by using T_{input} so that v_{input} is transformed to $v_{adjusted}$ (Figure 6). The line segment connecting v_{input}, and v_{rest} must intersect the boundary of the cone at $v_{adjusted}$. Using $v_{adjusted}$, rotation angles α (for first DOF of the joint) and β (for second DOF of the joint) from the rotation components are newly calculated to produce the adjusted rotation matrix $T_{adjusted}$.

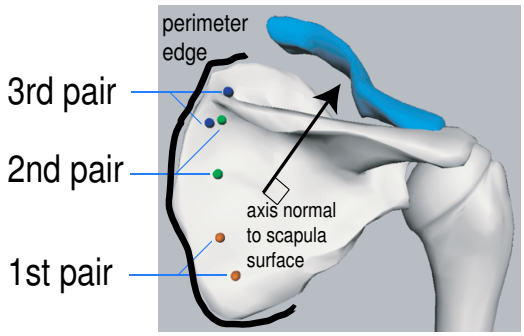

Figure 7: Reference points on scapula bone

4.7 Scapula Constraint Component

The *scapula constraint component* is a custom-designed component to handle the specific situation of the scapulo-thoracic constraint in the shoulder. This example illustrates how the component framework can be extended for special handling of an individual joint. The scapula bone is always gliding on a curved surface defined by ribs, muscles and fatty structures. To represent this in our model, we use an ellipsoidal constraint as others have done in the past [Garner and Pandy 1999; Maurel and Thalmann 2000]. However, instead of using only one ellipsoid, we have chosen to use two, with one for each side of the rib cage (Plate 6). This allows the constraints on both sides of the rib cage to be properly maintained as the spine is twisted or laterally bent. In order to constrain the scapula bone to be gliding on the surface of an ellipsoid, we define pairs of reference points on the scapula, and make sure that at least one active pair stays on the ellipsoid at all times (see Figure 7).

We describe how the DOF are fully determined for the scapula. Let us think of the scapula as an initially free joint with three DOF for rotation and three DOF for translation. Because it is attached to the parent clavicle bone, the three DOF of translation are determined by its shared attachment point with the clavicle. We further constrain the rotation of the scapula around an axis perpendicular to its flat surface to be dependent on the abduction of the humerus bone using a dependency component, removing another DOF. The ellipsoidal constraint determines the remaining two DOF or rotation. By constraining a pair of reference points on the scapula to the ellipsoid's surface, the configuration of scapula bone is fully determined.

We define our pairs of reference points to lie near the outer perimeter of the scapula as shown in Figure 7. Having several pairs of reference points allows the contact area between the scapula and rib cage to change depending on other joints in the shoulder. Referring to Figure 7, the area close to the 1st pair is more likely to be in contact with the rib cage when the shoulder is lifted. The 2nd pair is more likely to be in contact when the shoulder is lowered. The 3rd pair is active when the scapula is fully rotated clockwise around the axis normal to its surface. Therefore, these three pairs of reference points are used to find a new contact pair by interpo-

lating over two DOFs corresponding to the amount of shoulder lift and rotation about the scapula.

The scapula will then be rotated twice to constrain these contact points on the surface of the ellipsoid. In the first rotation, a predefined vector x_1 going through the joint origin is used as the rotation axis. Rotation of an angle θ around x_1 will bring the first reference point onto the ellipsoid surface. In the second rotation, the vector connecting the joint origin and the first reference point is used as the rotation axis, x_2. Similarly, rotation of a angle ψ around x_2 will bring the second point onto the ellipsoid. Notice that the second rotation will not change the position of the first contact point since it is on the rotation axis x_2. Binary search is used to find both rotation angles θ and ψ.

In general, it is desirable to create joint components that can be reused. Nevertheless, the ability to create very specialized constraints can be useful to create tailored, intuitive parameters to simplify the description of complex articulations unique to a particular joint. More biomechanical detail can be added to a joint component as deemed necessary for the application.

5 Results

Having described all the joint components, we can connect them in a network to construct joint set functions for the segments of our skeleton. We will describe two cases of fairly complex joints we created using our framework: the spine and the shoulder. Initial estimates of parameter data for the joints were determined from literature on joint physiology [Kapandji 1982b; Kapandji 1982a]. Custom plug-ins were developed for the Maya 3-D modelling software [Alias/Wavefront 1999] to allow interactive placement of the bones and adjustment of joint parameters. Maya's advanced modelling environment allowed articulation of joint sets to be evaluated interactively. Once we were satisfied with the joint model, we exported the parameters for all the joint components in an XML-based file format which can be loaded into our own OpenGL-based custom application software. We implemented our joint component framework in these two different software environments to test our ability to interchange joint models between them. We can achieve interactive rates on a Pentium III 933 MHz Windows 2000 machine, with a Nvidia GeForce4 graphics card.

5.1 The Spine

There are twenty-four movable vertebrae in the spine of a human. According to their position and functionality, they are divided into three joint sets: the cervical region (seven vertebrae in the neck), the thoracic region (twelve vertebrae in the thorax), and the lumbar region (five vertebrae in the abdomen) [Kapandji 1982a]. For the thoracic joint group, we also included all the ribs and the sternum, creating the rib cage. For all three spine joint sets, the same type of joint function is used. The difference between them is just the joint parameters given for each joint group, where the amount of rotation in the thoracic vertebrae is considerably less than the cervical and lumbar regions.

For example, the cervical joint set has seven joints (c1-c7) as well as seven bones (including both the vertebrae and the discs between any two vertebrae). Each joint alone has three DOF of rotation and thus has three rotation components. Rotation axes and rotation centers are estimated from [Kapandji 1982a] for each rotation component. The three rotation components for a single joint may have quite different rotation centers and non-orthogonal rotation axes. A pair of joint limit angles defined by a one-to-many mapping component is provided to bound each of the rotations. Since rotation behavior of the vertebrae in the spine are coupled together, we simplify movement control to have only three DOF: flexion/extension, lateral-bending, and twisting along the vertebra

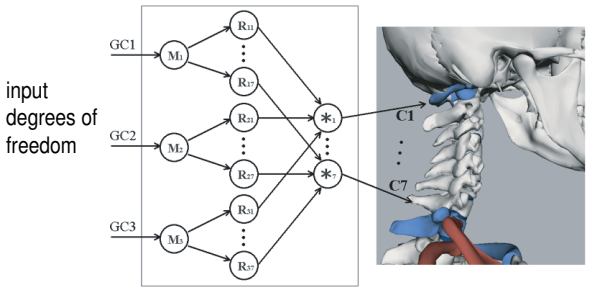

Figure 8: Joint function of cervical joint set (Note: M: one-to-many mapping, C: compensation, R: rotation, D: dependency, L: joint cone limit, S: scapula constraint, *: multiplication)

axis. In each joint set of the spine, a one-to-many mapping component first converts the input DOF to a rotation angle for each vertebra in the joint set.

For example, in the cervical region, the flexion/extension maps to the following angle rotation ranges (in degrees) for the seven vertebrae (refer to Figure 8): C7 [-13.2,5.5], C6 [-7.5,5.5], C5[-4,5.5], C4[-4.6,5.5],C3[-8,6.5],C2[-5.5,6.5],C1[-18.5,6.5]. The conversion is a linear map between [-1,1] and a pair of vertebra-specific angle limits [min, max]. Each angle is then sent to a specific rotation component for the vertebra it corresponds to.

The output matrices of the rotation components for each DOF will be directed into a matrix multiplication component to generate the final transformation for the vertebrae. This is illustrated in Figure 8 for the cervical joint set.

The implementation of joint set functions for the thoracic and lumbar are similar to the cervical vertebrae with the exception that the thoracic group contains ribs attached to each vertebra. Because the rib cage creates a closed chain with the spine and sternum, it tends to resist thoracic spine movement that would otherwise cause the individual ribs to rotate away or into each other during lateral bending. By carefully choosing the joint parameters for the ribs, the rotation of a rib can be set to be dependent on the amount of motion of its attached thoracic vertebra to maintain the overall shape of the rib cage. Intuitively, we should rotate the ribs in a direction opposite to that of the spine's rotation with the ribs always rotating less than spine. Therefore, we choose to define the axes of ribs to be opposite to those for vertebrae and define their rotation limits to be smaller. As shown in Plate 2, we can successfully approximate the non-rigidity of the whole rib cage (but not that of single ribs). For more accurate deformations of the rib cage, a custom joint component can be designed.

To summarize, our spine model is composed of three joint sets. Each joint set has three DOF of rotation for flexion/extension, lateral bending, and twist, making a total of nine DOF to control the entire spine and the rib cage. This is considerably less than the total number of articulations achievable in our model because we have implicitly built in the various dependencies. Although our model is probably still not as accurate as a real human spine, we can achieve fairly realistic spine configurations (Plates 3 & 4) with a lightweight, intuitive control mechanism. We believe that our spine model is an acceptable compromise between the need for accuracy and simplicity of control.

5.2 The Shoulder

The shoulder is one of the most complex joints in the human body, making it a good test of the versatility of our joint component framework model. The shoulder comprises four articulations (the scapulo-humeral joint, the acromio-clavicular joint, the sterno-

clavicular joint, and the sliding scapulo-thoracic joint) (Figure 2). Several bones are involved in these joints: the clavicle, scapula, humerus, sternum and rib cage (Plate 5). In addition to the bone articulations, ligaments, cartilage and muscles also play important roles in the shoulder to create coupling behavior and dependencies among the shoulder's joints.

We put the shoulder complex (four joints and three bones) into a shoulder joint set. Its joint set function has five DOF, of which three control the scapulo-humeral joint (flexion/extension, abduction/adduction and twist of the humerus bone) and two control the sterno-clavicular joint (vertical and horizontal rotation of the clavicle bone). The acromio-clavicular joint has zero controllable DOF because its movement can be fully determined by movements of the other two joints and the ellipsoidal surface constraint with the rib cage. The joint set function outputs three transformation matrices for the three joints respectively. Inside, the function has three parts, each in charge of one joint which we now describe.

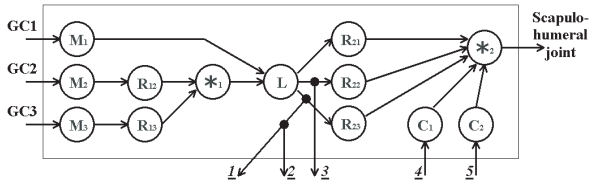

Figure 9: Joint function of shoulder joint set: part 1

The first function part computes the matrix for the scapulo-humeral joint. Shown in Figure 9, each control input goes first through a one-to-many mapping component at which time it is converted to an angle in the range of [-180, 180]. The two angles of flexion/extension and abduction/adduction will be sent into two rotation components to create their rotation matrices. For the right arm abduction/adduction, the rotation component has two intervals, [0,60] and [60,180] degrees, with rotation centers in the humerus local coordinate system of (-0.373, -0.247, -0.181) and (-0.373, 0.449, -0.181) respectively.[1] The combined result of these two matrices, together with the twist angle, will then go through a joint cone component (Figure 5) to produce a final valid orientation matrix. As we need to compensate movements from both the parent joint and grandparent joint of scapulo-humeral joint, two compensation components are needed. Their inputs are the final outputs of the other two function parts, which will be discussed shortly. By multiplying compensation matrices and the final rotation matrices together, we get the final transformation of this joint.

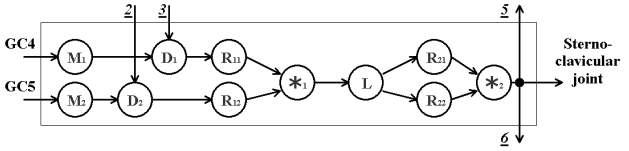

Figure 10: Joint function of shoulder joint set: part 2

Proceeding to the next function part, which is in charge of the sterno-clavicular joint (Figure 10). Each input DOF goes through a one-to-many mapping component and gets converted to an angle. The two angles, together with two angles of the scapulo-humeral joint, are sent as input into two dependency components since the rotation angles of the clavicle partially depend on those of the upper arm as we described in Section 4.5. For example, in the right shoulder, a piecewise linear function maps the humerus abduction/adduction angles of 0, 60, 120, 180 degrees to the clavicle elevation angles of -10, -6.7, 4.85, and 23 degrees respectively. Rotation components construct matrices using the outputs from dependency components and the combined result gets tested in a joint cone component. Its output provides the final transformation.

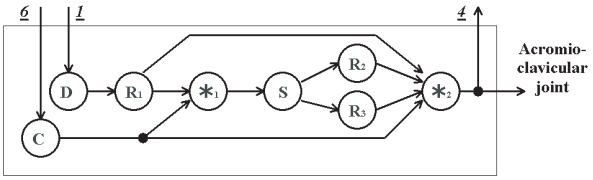

Figure 11: Joint function of shoulder joint set: part 3

The third function part controls the acromio-clavicular joint (Figure 11). All its inputs are from the other two function parts discussed above. From the first part, we have the abduction angle of the upper arm, which will decide, through a dependency component and a rotation component, the rotation of the scapula bone around the axis normal to its flat surface. The second part provides the transformation of the sterno-clavicular joint, which will be used to compute the compensation matrix. Then the scapula constraint component will use the combined results to produce another two angles. For the right shoulder, the initial local axis of rotation on the scapula was (0.263, -0.912, 0.314). The three pairs of reference points on the scapula (see Figure 7) in its local coordinates were (8.5,-4,-0.3) with (8.7, -2.4, -0.1), (8.2,0,-0.1) with (7.5,2,0), and (8,2,-0.8) with (7,3,-0.6) respectively. The two angles are used to rotate the scapula onto the surface of an ellipsoid that is fixed in the sternum's frame of reference. Finally, the three rotation matrices and the compensation matrix are multiplied together, bringing us the final transformation.

Notice that these three function parts are tightly linked together. It is not possible to completely compute any of them independently of the others. This is because the computation of the first part depends on the final results of the other two parts, which need intermediate results from the first part as their inputs. However, there exists a strict topological order for these computations over the whole network of these three parts, allowing the joint function to be serially computed with no danger of deadlock.

To summarize, the shoulder in our model is deterministically controlled by a complex function with a very simple interface (only five DOF). Realism is not sacrificed. Not only do we model the independent movements of the shoulder and upper arm, their coupling behavior is captured as well. (Plates 5 & 6)

6 Conclusion

In this paper, we introduced a new method for building joint models for articulated figures. We develop a repertoire of different joint components, each of which implements a simple functionality yet combined can produce complex behaviors. The interface inputs to the joint set functions can be designed to have intuitive interpretations to an animator, and be sufficiently compact because all dependencies and coupling of the joints are implicitly built into the joint set function. Joint components can be reused in different contexts or extended to incorporate increasing accuracy in the joint model.

We have filled our repertoire with seven components. Using these components, we outlined the construction of two complex joints for the upper body in our skeleton model. They each have a simple control interface, and can be computed quickly enough for

[1] These coordinates depend on the measurement units, bone geometry and local coordinate systems of the segments used to represent the skeleton model and therefore will vary between different models.

real-time interactive applications, while simultaneously producing realistic joint movements. On the other hand, our joint models are not an exact replica of real anatomical joints. The joint designer is enabled through the framework to consider different competing criteria such as desired level of detail, joint parameter design, and computational complexity to develop suitable models for the desired target application.

Due to the complexity of several joint functions, it may not always be possible to compute their analytic derivatives which would be needed for inverse kinematics or gradient-based optimization. However, as the joint set functions produce deterministic joint transformations, one could use finite difference techniques to estimate their derivatives. We are currently using this framework to find subject-specific parameters that will allow the same joint set functions to be customized for different individuals or animals. The seven joint components we defined do not represent a complete set for modelling all joints at every level of accuracy. In particular, through the design of new components or new joint sets, we would like to incorporate and combine various joint models developed in the biomechanics community to make them easily accessible within a single convenient toolbox.

Acknowledgments

This project is supported by the Honda Research Institute USA, Inc. We would also like to thank the reviewers for their helpful suggestions.

References

ALIAS/WAVEFRONT, A. D. O. S., 1999. Maya. 3-D modeling system.

BADLER, N. I., PHILLIPS, C. B., AND WEBBER, B. L. 1992. *Virtual humans and simulated agents*. Oxford University Press.

BULL, A. M. J., AND AMIS, A. A. 1998. Knee joint motion: description and measurement. In *Proc. Instn. Mech. Engrs.*, vol. 212 Part H, 357–372.

CRAIG, J. J. 1989. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley.

DELP, S. L., AND LOAN, J. P. 1995. A graphics-based software system to develop and analyze models of musculoskeletal structures. *Comput. Biol. Med. 25*, 1, 21–34.

DEROSE, T., KASS, M., AND TRUONG, T. 1998. Subdivision surfaces in character animation. In *Computer Graphics (SIGGRAPH '98 Proceedings)*, 85–94.

GARNER, B. A., AND PANDY, M. G. 1999. A kinematic model of the upper limb based on the visible human project (vhp) image dataset. *Computer Methods in Biomechanics and Biomedical Engineering 2*, 107–124.

GIRARD, M., AND MACIEJEWSKI, A. A. 1985. Computational modeling for the computer animation of legged figures. In *Computer Graphics (SIGGRAPH '85 Proceedings)*, B. A. Barsky, Ed., vol. 19, 263–270.

GRASSIA, F. S. 1998. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools 3*, 3, 29–48.

HODGINS, J. K., O'BRIEN, J. F., AND TUMBLIN, J. 1998. Perception of human motion with different geometric models. *IEEE Transactions on Visualization and Computer Graphics 4*, 4, 307–316.

HUMANOID ANIMATION WORKING GROUP, 1999. H-anim 1.1 specification for a standard humanoid. www.h-anim.org.

KAPANDJI, I. A. 1982. *The Physiology of the Joints: The Trunk and the Vertebral Column*, 2 ed., vol. 3. Churchill Livingstone.

KAPANDJI, I. A. 1982. *The Physiology of the Joints: Upper Limb*, 2 ed., vol. 1. Churchill Livingstone.

LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Computer Graphics (SIGGRAPH 2000 Proceedings)*, 165–172.

MACIEL, A., NEDEL, L. P., AND DAL SASSO FREITAS, C. M. 2002. Anatomy-based joint models for virtual human skeletons. In *Proceedings of Computer Animation 2002*, 220–224.

MAUREL, W., AND THALMANN, D. 2000. Human shoulder modeling including scapulo-thoracic constraint and joint sinus cones. *Computers and Graphics 24*, 2, 203–218.

MONHEIT, G., AND BADLER, N. I. 1991. A kinematic model of the human spine and torso. *IEEE Computer Graphics and Applications 11*, 2 (March), 29–38.

SCHEEPERS, F., PARENT, R. E., CARLSON, W., AND MAY, S. F. 1997. Anatomy-based modeling of the human musculature. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, 163–172.

SHOEMAKE, K., AND DUFF, T. 1992. Matrix animation and polar decomposition. In *Proceedings of Graphics Interface '92*, 258–264.

SHOEMAKE, K. 1985. Animating rotations with quaternion curves. In *Computer Graphics (SIGGRAPH '85 Proceedings)*, vol. 19, 245–254.

SLOAN, P., ROSE, C., AND COHEN, M. 2001. Shape by example. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, 135–143.

TOST, D., AND PUEYO, X. 1988. Human body animation: a survey. *The Visual Computer 3*, 5 (March), 254–264.

WANG, X. C., AND PHILLIPS, C. 2002. Multi-weight enveloping: Least-squares approximation techniques for skin animation. In *2002 ACM SIGGRAPH Symposium on Computer Animation*, 129–138.

WILHELMS, J., AND VAN GELDER, A. 1997. Anatomically based modeling. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, 173–180.

WILHELMS, J., AND VAN GELDER, A. 2001. Fast and easy reach-cone joint limits. *Journal of Graphics Tools 6*, 2, 27–41.

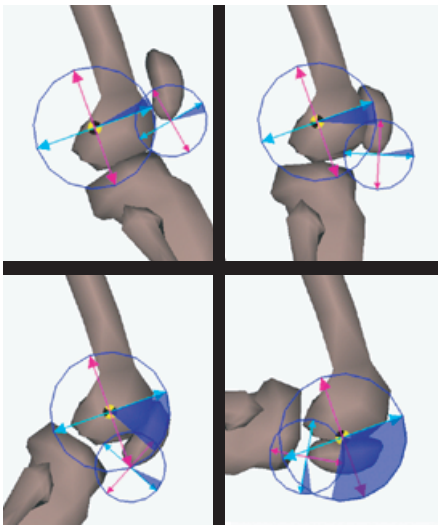

Plate 1: Four snapshots of the knee joint driven by a single DOF. Notice how the joint center (yellow and black disk) changes during the motion.

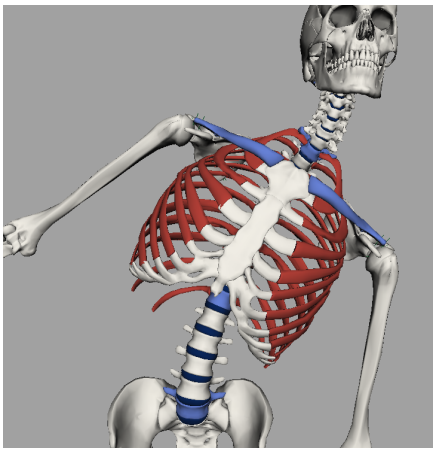

Plate 2: Lateral bending in the spine with the rib cage intact

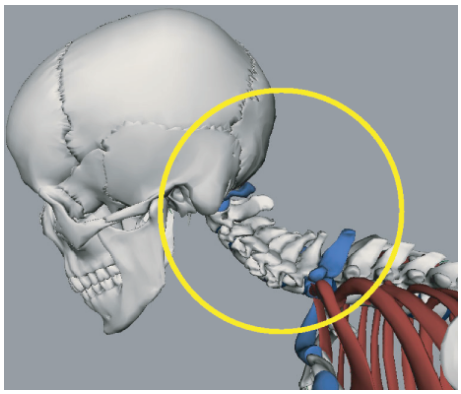

Plate 3: Flexion in cervical vertebrae (within yellow circle)

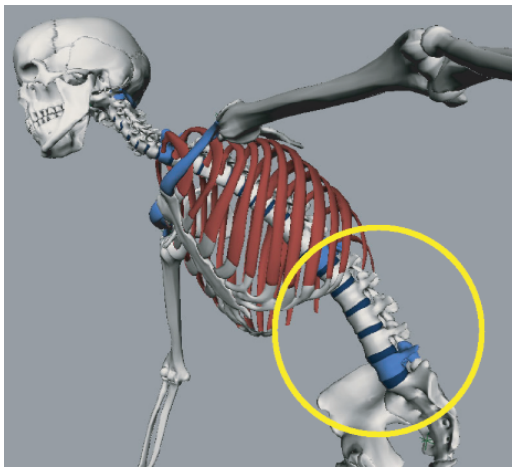

Plate 4: Lumbar region of spine (within yellow circle)

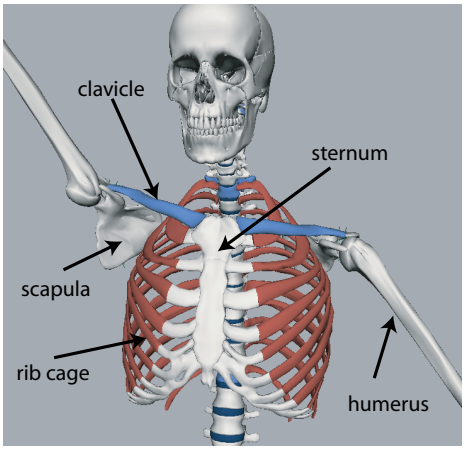

Plate 5: Extreme abduction of right arm in shoulder

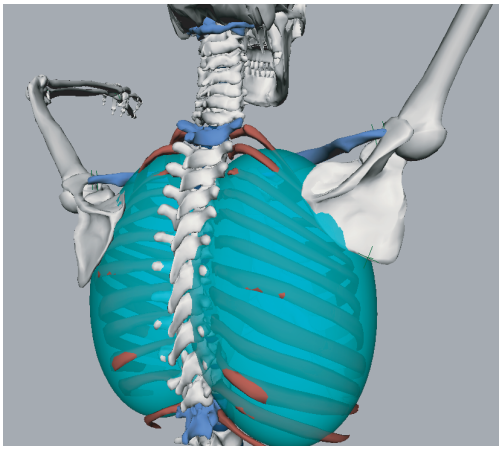

Plate 6: Ellipsoids to constrain scapulo-thoracic joint