A Musculoskeletal Model of the Upper Limb for Real Time Interaction

Amos Patrick, Karim Abdel-Malek,
Virtual Soldier Research (VSR) Program, University of Iowa

ABSTRACT

With the ever-increasing power of real time graphics and computational ability of desktop computers, the desire for a real-time simulation of the musculoskeletal system has become more pronounced. It is important that this simulation is realistic, interactive, runs in real time, and looks realistic, especially in our climate of Hollywood special-effects and stunning video games. An effective simulation of the musculoskeletal system hinges on three key features: accurate modeling of kinematic movement, realistic modeling of the muscle attachment points, and determining the direction of the forces applied at the points. By taking known information about the musculoskeletal system and applying it in a real time environment, we have created such a model of the human arm. This model includes realistic constraints on the joints and real-time wrapping algorithms for muscle action lines. Preliminary evaluation shows that the moment arms calculated by our model are similar to those shown in the literature. Furthermore, by coupling our model with known optimization algorithms, muscle activation levels for prescribed joint torques can be calculated in real time.

INTRODUCTION

The aim of virtual human development is to create a digital representation of a living human which exists within a computer. This tool, which is based on anatomically accurate models and realistic actions, will provide valuable feedback for designers, engineers, scientists, and doctors regarding how people will react or interact with new products, different environments, or anything else which can be presented in a digital form. While a complete representation of a human in a digital world is still a ways off there are currently significant advances being made in the development of interactive real time virtual humans. This is due to the convergence and development of several areas of knowledge, primarily engineering, gaming, and computer science.

Development of a musculoskeletal model has been around for quite a while. Mechanical models of the shoulder started out as simplified two-dimensional models that simply analyzed humeral movement with respect to a non-moving scapula (DeLuca and Forrest, 1973; Poppins and Walker, 1978). About the same time Dvir and Berme as well as Jackson et al also published literature that presented shoulder models, which were also confined to single motion patterns (Dvir and Berme, 1978; Jackson et al, 1977). These model were expanded to three degrees of freedom and improved upon throughout the 1980's and early 90's.

Up until the mid '90 all the models presented were essentially mathematical models that were not concerned with representing the musculoskeletal system as a realistic 3D geometric model. However, with the advances medical imaging technology, accurate models of skeletal system were become more readily available as well as the computer graphics to display them. In 1995 Delp and Loan published a paper where they presented a software tool called SIMM that allowed users to create, modify, and evaluate various musculoskeletal modes (Delp and Loan, 1995). With this software the user was allowed to graphically interact with the skeleton.

A year later Maurel et al. presented another model of the upper limb that included a realistic graphical representation of the arm specifically designed for dynamic simulation (Maurel et al, 1996). This model was created as part of the European Esprit Project CHARM. This paper was followed up by another paper by Maurel and Thalmann 1999 where a case study of modeling the arm was presented (Maurel and Thalmann, 1999).

Today there are several commercial musculoskeletal models on the market. These include SIMM, VIMS, AnyBody, and LifeMOD. One of the key differences between the model presented in this paper and the ones currently available is that our model is focused on real time interaction. By this we mean that there is no obvious delay between user input and the model's output. Furthermore, these software tools are not used extensively by ergonomists in industry.

There are currently three widely used commercial avatars on the market. These include Jack from UGS,
Ramsis from Human Solutions, and Safework. While these avatars are widely used for ergonomic analysis, none have incorporated a musculoskeletal system within their avatar for real time analysis of muscle forces.

Our goal is to incorporate an accurate musculoskeletal system within Santos™, the avatar currently being developed within VSR. This system will allow for real time interaction and analysis of muscle forces within a package which is designed for widespread use. This paper will present our formulation of the upper arm. This same formulation will later be applied to the rest of the body.

3D MODEL

The objective is to create a real time three dimensional model of the musculoskeletal system which accurately reflects the complex interaction of the muscles working together to create torque about a joint. In order to develop a general method for creating this model we decided to focus on the upper limb. This was chosen for two reasons. First for the complexity of the shoulder girdle which will ensure the development of a robust formulation. Second, this model will eventually be incorporated into Santos environment which is currently using extensive upper body movement.

A successful model will hinge on two factors, appropriate modeling of the kinematic chain of the bones and the attachment as well as the direction of force for all the muscles. These factors are important because the primary purpose of this model is to obtain the relationship between muscle forces and torques. An accurate kinematic model is important because if joints rotate about an inaccurate point or axis, the resulting torque-force relationship will be flawed. Similarly, if the muscles are not attached in appropriate positions or the direction of the force applied to the bone is not pointing in a realistic manner, the relationship is again compromised. Therefore it is paramount that these fundamental issues be carefully addressed.

KINEMATIC MODEL

The kinematic chain of the upper arm is composed of five bones as shown in Figure 1. It should be noted that for our model the hand was considered a rigid extension of the ulna.

These bodies were arranged into a parent-child hierarchy shown in Figure 2.

Figure 1: Bones of the right arm

Figure 2: Hierarchy of the bones in the arm

Figure 3: Joints of the arm

Between each bone a joint had to be defined. In a true human joint, most movement is a combination of rolling and sliding between the bones (Engin, 1984), thus no fixed axis or center of rotation can be found. However, this movement is generally considered negligible for the arm, so the joints are modeled either as an ideal ball and socket joint with three degrees of freedom or a hinge joint with one degree of freedom. Following previous kinematic arm models (Maurel and Thalmann, 1999; Engin, 1980; Helm, 1994; Hogfors et al, 1987; Raikova,
1992) the sterno-clavicular (SC), acromio-clavicular (AC), and gleno-humeral (GH) joints are modeled as ball and sockets while the ulno-humeral (UH) and ulno-radial (UR) are modeled as hinge joints (Figure 3).

Rotations in joints of more than one degree of freedom are handled similar the model presented by Maurel et al where rotations will be described as Euler angles defined to rotate in the following order: $\theta$ about $z$, $\phi$ about $y$, and $\psi$ about $x$ (Maurel et al., 1996). To accomplish this in our simulation instead of using a single local frame, three frames are stacked on top of each other. Then to achieve a specified orientation the first frame is rotated about it’s $z$-axis by $\theta$, the second frame is rotated about it’s $y$ axis by $\phi$, and frame three is rotated about it’s $x$ axis by $\psi$.

One of the more difficult features about modeling the shoulder is how to treat the scapula and its interaction with the thorax. The scapula is located between the clavicle and the humerus in the kinematic chain and is constrained to glide across the thorax (Dvir and Berme, 1978). Because the scapula is nearly completely surrounded by soft tissues, there are no articular structures between the scapula and the thorax, resulting in the scapula being able to move in all six degrees of freedom. Since the scapula glides across the thorax two forms have been proposed to describe this constraint, a dot contact or a linear contact (Maurel and Thalmann, 1999). A dot constraint describes a situation when one point of an object is constrained to a surface, thus allowing 5DOF. A line constraint describes a situation when two points of an object are constrained to surface resulting in 4DOF. It was decided that the linear contact constraint is more representative of movement of the shoulder as described by Dvir and Berme but instead of using an idealized ellipsoidal thorax as van der Helm did we used the actual geometric model of the thorax (Helm, 1994).

To accomplish this we started by first location two coordinate frames on the medial edge of the scapula with their $x$-axis pointing towards the thorax. These points will be called g1 and g2 (Figure 4). A ray is cast along each of the frames $x$-axis to determine the distance between the frame and the thorax. Another coordinate system is created at the SC joint such that its $z$ axis points towards g1 and the $x$ axis is found from the cross product between this $z$ axis and another vector pointing toward g2. Then to keep the scapula on the thorax, it is rotated about the SCy until the distance between g1 and the thorax is within a certain range. Next the scapula is rotated about SCz until the distance between g2 and the thorax is within a certain distance.

MUSCLE MODEL

As noted in the previous section, there are five joints in our model of the upper extremity. Crossing these five joints are 21 major muscles. The common method used to model muscles is to idealize them as one or more action lines which represent the path along which the tension of the muscle is traveling. We chose to use this method because it is a computationally efficient way of representing the forces generated by the muscles.

When attaching the action lines to the skeleton, there are three factors which are important: the placement of the insertion and origin, the number of action lines used, and how the action line wraps over underlying structures.

Each muscle attached to the skeleton in two or more places. The most distal attachment point is usually referred to as the insertion point while the muscle origin is proximal. In reviewing the literature one can find many studies which attempt to map the location of these points in space. However, applying these locations to a different study usually proves to be problematic because of issues with posture, body size, and coordinate systems. Therefore, we decided to resolve these positions based on anatomical landmarks and highly detailed polygonal models of the musculoskeletal system. These polygonal were created.

The second factor that must be considered when attaching the muscles to the skeletal model is the number of action lines used to approximate the force generated by a muscle. For computational reasons, the muscles will need to be idealized as a number of action lines along which the tension being created by the muscle will act. Obviously, the more action lines used, the more accurate the model will become. However, for the model to be useful and to fulfill the requirement of being real time, a balance between computational
expense and accuracy must be found. For many muscles a single action line is adequate because the origin and insertion points are small, such as the biceps. For other muscles, such as the trapezius, the insertion and/or origin cover a wide area so a single action line is not an appropriate approximation (Figure 5). Van der Helm and Veenbaas compared the moment vector calculated from a large number of action lines (200) and compared it to the moment vector calculated from six action lines and found the error to be minimal (Helm and Veenbaas, 1991). Our model however will be developed similar to the model presented by Maurel and Thalmann in which the number of action lines will be less (Maurel and Thalmann, 1999). Once this model is finished, the real time performance will be evaluated and more action lines can be considered.

Figure 5: Trapezius and its action lines

Figure 6: Straight line action model for the short head of the bicep. Note that this model results in the action line passing through the distal end of the humerus when the arm is fully extended

Many muscles can never be modeled as a straight line from origin to insertion. For example, consider the deltoid, which originates on the scapula and inserts on the humerus after curving over the gleno-humeral joint. Because of the curved nature of this muscle, representing the action line as one or more straight lines would be insufficient.

The centroid line method is another approach that can be used to model the action lines of muscles (Jensen and Davy, 1975). It does not assume the force runs in a straight line between known attachment points. Instead, it predicts the lines of force can be considered to be acting along the centroid of the transverse cross section of the muscle. The unit vector of the action line predicted by each of these methods is significantly different (An et al, 1981 and Mikosz et al, 1988). While this method creates a more accurate action line, it is not very useful for our model because it only represents action lines of muscles in a fixed orientation.

A third method is used to compromise between the simplicity of the straight-line approach and the accurate but static contour approach. This method uses a line that
has one end attached to the origin point and the other attached to the insertion point. However, instead of a simple straight line, the line is allowed to wrap around a number of obstacles through the use of via points, which represent underlying muscles and bones. Typically, these via points are fixed to a bone during movement. For example, consider the triceps, which wrap around the distal end of the humerus when the elbow is flexed (Figure 7). A set of reasonable via points for modeling the force line of this muscle would be fixed with respect to the humerus, thus keeping the action line out of the end of the humerus. The only algorithm needed would be to determine which of the via points are active. None of the via points will be active when the elbow is fully extended and all the via points will be active when the elbow is fully flexed.

![Figure 8: Lower action line of the trapezius with fixed via points on the medial edge of the scapula. Note how these fixed via points skew the line when the scapula is moved](image)

While this may work well for muscles that only crosses a joint with only one degree of rotational freedom, fixed via points become a problem when more complex wrapping must occur. For example, consider the action line shown in Figure 8 which represents the lower action line of the trapezius. This line originates in the thoracic region of the spine, wraps around the thorax and the posterior surface of the scapula. The movement of the scapula and shoulder affects the shape of this muscle. If one were to use fixed via points the action line would not be allowed to slide over the medial edge of the scapula during movement of the shoulder, thus producing an inaccurate approximation of the muscles centroid line.

REAL TIME MUSCLE WRAPPING AND SLIDING

This section discuss the method used for real time approximation of how muscles will wrap and slide around underlying structures by allowing floating via points to dynamically wrap and slide prescribed obstacles. The difference between the floating via points and the fixed via points is that the floating via points are not fixed to a bone but are allowed to slide across the surface of the obstacle. This is the key difference between our approach and the method presented by Charlton and Johnson (Charlton and Johnson, 2001). Our approach makes two assumptions about the action lines. The first is that the action line or lines of a muscle can be modeled as a frictionless elastic string wrapping around prescribed obstacles. The second is that spheres and cylinders can represent the underlying anatomical structures that the muscle must wrap around. The lines being wrapped will then be used to approximate the force lines generated by the muscles.

PROBLEM FORMULATION

We propose an algorithm that uses prescribed obstacles to bend the action line of a muscle in real time such that the action line is a reasonable approximation of the real muscle’s action line for all possible joint configurations. From anatomy we can determine the origin and insertion of a particular muscle. We can also analyze underlying muscles and bones so as to determine the size, shape, position and number of obstacles that the muscle being modeled must wrap around. Therefore, the unknowns are the positions of the intermediate floating via points. These positions will be dependent on two factors, if there are one or more obstacles about which wrapping should occur and, if so, the shape and position of the obstacle or obstacles.

PROPOSED SOLUTION

An action line will be used to idealize the direction of force generated by a single muscle. The model attempts to use obstacles to push the line so that it approximates the centroid line of the muscle, regardless of the orientation of the arm. Some muscles with wide attachment point or multiple heads such as the bicep use multiple action lines. All action lines consist of an origin point at one end, an insertion point at the other end and a number of floating via points in between. Each point is connected to the next with a straight line. An obstacle consists of a simple sphere or cylinder; therefore wrapping algorithms will need to be developed for each obstacle.

WRAPPING ALGORITHM FOR A SPHERE

To present this algorithm we will assume there is only one obstacle, which is a sphere, and the action line will consist of seven points, which will be referred to as number 1 through 7, according to their position on the line. Point 1 and 7, which are the two ends of the line, will always be fixed to the origin and insertion, respectively. Points 2 through 6 are floating via points, which slide across the surface of the sphere when wrapping.

The first step of the algorithm is to determine if wrapping should occur. Since the action line is being idealized as a frictionless elastic string, it will always move to the position where it has the lowest potential energy. As gravity is being ignored, this translates into the shortest path from the origin to the insertion. Therefore, wrapping
will only occur if the sphere is between the origin and the insertion. This is determined by casting a ray from the origin to the insertion and testing to see if it intersects with the sphere. If no intersection is detected, then no wrapping will occur. In this case, the five intermediate via points are evenly spaced along the line from the origin to the insertion.

If an intersection occurs then the line will need to be wrapped around the sphere. The line will still be along the shortest path between the origin and insertion while wrapping around the sphere. For most cases, this path will lie on a plane that contains the insertion, origin and the center of the sphere. We will refer to this plane as the action line plane, or AL plane. The exception to this is when the origin, insertion, and center of the sphere are collinear. In this situation, all paths around the sphere are the same distance so there is no unique solution. Fortunately this rarely happens and can be avoided by slightly shifting one of the points.

Once the AL plane has been defined, determining the position of the floating via points becomes a two dimensional problem. Furthermore, if the sphere’s z axis is aligned with the normal vector of the plane, then these positions can be determine in the x-y plane of the circle, where the center of the circle is the origin of the circles coordinate system (Figure 9).

In this algorithm, the first two points determined are the number 2 and 6. These points are where the line first contacts the sphere (point 2) and stops contacting the sphere (point 6). We will call these points $T_o$ (origin side tangent) and $T_i$ (insertion side tangent). The method we use to calculate these points is similar to that presented by Charlton and Johnson except we calculate $\theta$ with (1) (Charlton and Johnson, 2001).

$$\theta = \sin^{-1}\left(\frac{T_i \cdot T_o}{|T_i||T_o|}\right)$$ (1)

This allows us to avoid having to calculate $f$, $\beta_1$, and $\beta_2$ as shown in (Charlton and Johnson, 2001). Now that we have determined the location of points 2 and 6, we need to calculate the location of points 3, 4 and 5. These will simply be evenly spaced along the arc created on the surface of the sphere between $T_o$ and $T_i$.

The final step for this algorithm is to transform all the points from the sphere coordinate system to the world coordinate system then set the curve point positions to the appropriate point. Point 1 has already been set to the origin position, point 2 is set to $T_o$, point 3 is set to $\sqrt{3}T_1$, point 4 is set to $\sqrt{2}T_2$, point 5 is set to $\sqrt{3}T_3$, point 6 is set to $T_i$, and point 7 is set to the insertion position.

WRAPPING ALGORITHM FOR A CYLINDER

To present this algorithm we will again assume there is only one obstacle, which is a cylinder, and the action line will consist of seven points, which will be referred to as number 1 through 7, according to their position on the line. Point 1 and 7, which are the two ends of the line, will always be fixed to the origin and insertion, respectively. Points 2 through 6 are floating via points, which slide across the surface of the cylinder if wrapping occurs. Testing for wrapping about the cylinder is more complicated than testing for wrapping about a sphere since a simple ray intersection does not indicates wrapping. The cylinder does not need to be between the insertion and origin for wrapping to occur. Therefore, testing for wrapping will have to occur later in the algorithm when more information known.

Since the action line is being idealized as a frictionless elastic string, it will always move to the position where it has the lowest potential energy. As gravity is being ignored, this translates into the shortest path from the origin to the insertion while wrapping around the cylinder. The shortest path will be a straight line when viewed in the plane of the string. This plane can be thought of as a piece of paper that is arranged such that all the points along the line are touching the paper (Figure 10).

This paper would have one end on the origin then wrap around the cylinder in the same manner as the action line. The far end of the paper would be resting on the insertion. So, in the geometric coordinate system of the world (x y z) containing the cylinder, it will look like a plane that has been warped to bend around the cylinder and the action line, which is contained on that plane, will also be warped to wrap around the cylinder. However, in the coordinate system of the plane (u v), or if one were to unwrap the plane so it is flat, this line is not warped but a straight line from the origin to the insertion.
As with the sphere algorithm, this problem will be defined in the coordinate system of the cylinder, with the z vector pointing along the axis of the cylinder (Figure 11). The first two points determined are the number 2 and 6. These points are where the line first contacts the cylinder (point 2) and stops contacting the cylinder (point 6). We will call these points $T_o$ (origin side tangent) and $T_i$ (insertion side tangent). Furthermore we will denote the insertion point as $I$ and the origin as $O$, both of which are known with respect to the cylinder’s coordinate system.

We will first calculate the x and y position of $T_o$ and $T_i$ by only considering the x-y plane of the cylinder. One will note that method for calculating these points on the sphere will also work to determine the x-y position of the points in the radial plane of the cylinder. Therefore, the x-y component of $T_i$ and $T_o$ can be determined in the same way they were determined for the sphere. Again, the x-y position of the three via points will be evenly spaced along the arc created on the surface of the cylinder between $T_o$ and $T_i$.

At this point we have enough information to determine if wrapping should occur. To determine this we will check to see if the z component of the cross product between $T_{o,z}$ and $T_{i,z}$ is either positive or negative, depending on how the line should wrap (Figure 12).

Next, we need to calculate the z coordinate of the points. This coordinate is again calculated using the method presented in (Charlton and Johnson, 2001).

Now that we have determined the location of points 2 and 6, we need to calculate the location of points 3, 4 and 5. These will simply be evenly spaced along the arc created on the surface of the cylinder between $T_o$ and $T_i$.

The final step for this algorithm is to transform all the points from the cylinder coordinate system to the world coordinate system then set the curve point positions to the appropriate point. Point 1 has already been set to the origin position, point 2 is set to $T_o$, point 3 is set to $\overline{v_{p1}}$, point 4 is set to $\overline{v_{p2}}$, point 5 is set to $\overline{v_{p3}}$, point 6 is set to $\overline{T_i}$, and point 7 is set to the insertion position.

MULTIPLE OBSTACLE WRAPPING

Many of the action lines in the body will need to wrap around multiple obstacles to be able to closely approximate the appropriate path. In the previous two formulations it was assumed that only one obstacle was encountered. This is handled by simply running each algorithm in series in the order they are encountered and offsetting which curve points they affect. Before any of
The wrapping algorithms are run, point 1 of the curve is set to the origin and the last point on the curve is set to the insertion position. The only thing added to the above algorithms is an integer offset that the algorithm uses to determine which portion of the curve it works on and the insertion/origin of that portion. As an example, we will consider a line that starts at the origin before wrapping over a cylinder and then wraps around a sphere before terminating at the insertion point (Figure 13). This curve will have 12 points, one point for the origin, one point for the insertion, five points wrapping around the cylinder and five points wrapping around the sphere. These points will be referred to as cp_n, where n is the number indicating their order. The first step is to set the position of cp_1 to the origin and the position of cp_12 to the insertion. As far as the wrapping algorithms are concerned the curve only has seven points with the first point being the origin and the seventh point being the insertion.

Therefore, the algorithm will consider the origin to be the position of cp_1+offset and the insertion to be the position of cp_7+offset. At the end of the algorithm it will set the position of cp_2 to \( T_o \), set the position of cp_3 to \( 1 \times T_1 \), set the position of cp_4 to \( 2 \times T_2 \), set the position of cp_5 to \( 3 \times T_3 \) and set the position of cp_6 to \( 4 \times T_4 \). Since the cylinder is the first obstacle encountered when moving away from the origin, it’s algorithm will be run first with the offset value set to zero. So, it will see cp_1 as the origin and cp_7 as the insertion and then will set the position of cp_2 through cp_6 based on the algorithm. Next it will run the algorithm for the sphere with the offset set to 5. The sphere-wrapping algorithm will see cp_6 as the origin and cp_12 as the insertion and set the position of cp_7 through cp_11 based on the algorithm. It is important to note the interrelationship between the two obstacles. The first obstacle will always consider cp_7, which is the \( T_o \) point in the second algorithm, to be the insertion point when calculating the wrapping positions for cp_2 through cp_6. Then the second obstacle will always consider cp_6, which is the \( T_i \) point in the first algorithm, to be the origin when calculating the positions for cp_7 through cp_11. Therefore, the solution for multiple obstacles must be found iteratively.

**RESULTS**

**REALTIME WRAPPING**

There were two main criteria for the real time wrapping algorithm. The first is that it causes wrapping and sliding about one or more obstacles. The second requirement is that the wrapping occurs in real time. By this we mean that while the simulation is running any obstacle or end point of the line could be moved and the line should react by changing its wrapping with no apparent delay. It is important to note that this is where our model is significantly different that the model presented by (Charlton and Johnson, 2001). Previous models pre-calculate a discrete number of fixed via points and then determined which via points are active with a look up table based on joint angles. This makes movement about more that 1 DOF complicated. It also does not allow for real time changes in the obstacle dimensions or position. Our algorithm recalculates the position of all the via points at each render frame. This allows us to arbitrarily move, rotate or scale the obstacles as well as rotate the joints in any DOF.

To evaluate the real time performance we tested our algorithm with up to three objects. The results can be seen in Figure 14.

![Figure 14: Multiple obstacle wrapping](image)

With three objects to wrap around, the total execution time for the wrapping calculation is about 0.225 ms per render frame, which accounts for less than 1% of the render time. However, since multiple object wrapping is an iterative process it takes more than one frame for the wrapping to find the correct positions. To determine how many frames it takes for the solution to converge, the simulation used for the multiple body wrapping example was paused and one of the objects was moved. Then the simulation was advanced one frame at a time so that the number of frames before the line stopped moving could be counted. If only object was moved the line stopped obvious movement after four frames. This process was repeated with all three obstacles being displaced when the simulation was paused. Furthermore, the distance that the objects were displaced was varied to determine if that had an affect on convergence time. It was determined that distance does not significantly change the time to convergence.
but the number of objects moved does. The longest convergence time encountered was when all three objects were moved at the same time. 8 frames were counted before the line stopped obvious movement. If the simulation is running real time (~30 fps) this translates into about 0.25 seconds. It should also be noted that in all cases, the first frame displayed the most drastic change in position.

MUSCLOSKELETAL MODEL

Using the algorithms detailed earlier, muscle wrapping was incorporated into the musculoskeletal model by assigning one or more wrapping algorithms to an action line. An example of this can be seen in Figure 15.

After determining the insertion and origin points of all the muscles as well as the position of the related wrapping obstacles the resulting 3D model is shown in Figure 16.

Finally, the moment arms calculated by the model were checked to make sure the signs and the magnitudes were reasonable. The results are shown in Figure 19. In this figure, the four columns represent the four joints examined. Under each column is a graphic representing the joint associated with that joint as well as the coordinate system used. In each column is a series of bars indicating a qualitative measurement of the moment arm created by each muscle. The color of each bar represents the axis about which the moment arm is calculated. Bars extending to the left of the center line indicate a positive moment arm while bars extending to the right indicate negative.
From this figure it can be noted that the moment arms are all reasonable and have the appropriate sign. Furthermore, as the model is manipulated, the magnitude of the moments arms changed in an appropriate manner. This data was recorded for five of the muscle action lines and compared with previous studies. The result is shown in Figure 20. In order to compare the results, our model was scaled so it was similar in size to the model presented by Murray et al. This was accomplished by uniformly scaling our model until the length of the humerus was 30.3 cm as defined by Murray et al. It can be observed from the figure that the moment arms calculated by our model fit reasonably well with the data presented in previous studies, especially the bicep, brachioradialis, and pronator teres. The triceps, while maintaining a shape that is similar to the other data, seems to like it needs to be shifted down about 1 cm. Because the shape is fairly consistent with the previous studies this seems to indicate that the insertion point may need to be adjusted. The brachialis moment arm starts out within the range of the previous studies but then arc out too low at the end. Furthermore, its peak happens at about 80° while the others either peak out at the end of the flexion. This seems to indicate that the wrapping obstacles need to be adjusted.

From this figure it can be noted that the moment arms are all reasonable and have the appropriate sign. Furthermore, as the model is manipulated, the magnitude of the moments arms changed in an appropriate manner. This data was recorded for five of the muscle action lines and compared with previous studies. The result is shown in Figure 20. In order to compare the results, our model was scaled so it was similar in size to the model presented by Murray et al. This was accomplished by uniformly scaling our model until the length of the humerus was 30.3 cm as defined by Murray et al. It can be observed from the figure that the moment arms calculated by our model fit reasonably well with the data presented in previous studies, especially the bicep, brachioradialis, and pronator teres. The triceps, while maintaining a shape that is similar to the other data, seems to like it needs to be shifted down about 1 cm. Because the shape is fairly consistent with the previous studies this seems to indicate that the insertion point may need to be adjusted. The brachialis moment arm starts out within the range of the previous studies but then arc out too low at the end. Furthermore, its peak happens at about 80° while the others either peak out at the end of the flexion. This seems to indicate that the wrapping obstacles need to be adjusted.

It should be noted that for this preliminary validation, we only compared our model to other published musculoskeletal models. Further validation by comparing our model to actual subject data is needed before our model can be extensively used. This preliminary comparison indicates our method is good and if further comparisons do not match, it will simply be a matter of adjusting the obstacles and muscle attachment points.

**CONCLUSION**

This paper has addressed the modeling and simulation of the musculoskeletal system of the shoulder girdle and upper extremity by combining real time simulation methods, advanced CAD modeling techniques, and new
methods for characterizing muscle wrapping about anatomical structures.

It has been shown that real time simulation is effectively used to simulate kinematic movement of the upper limb while monitoring in real time moment arms for all the muscles crossing the sterno-clavicular, acromio-clavicular, gleno-humeral, ulna-humeral, and ulna-radial joints.

It has been shown that the system developed with 35 action lines and five bones with 7 DOF representing the shoulder girdle and upper extremity can be simulated in a 3D interactive visual environment in real time allowing a user to explore various muscle moment arms by changing joint angles.

It has been shown that significantly higher accuracy of muscle activation can be obtained by accounting for muscle wrapping around underlying anatomical structures. A new method for muscle wrapping was developed using floating via points, which has allowed for the action line of the muscle to be more accurately represented by the centroid line of the muscle.

Compared with experimental results published in literature, it was shown that our system yields results of the muscle moment arms are very consistent with published data.

REFERENCES

22. Seireg, A. and Avikar, R.J. (1973) “A mathematical model for the evaluation of forces in the lower extremities of the musculoskeletal system.” J. Biomechanics 6, 313-326
torques of human arm muscles as a function of elbow angle”, *J. Biomechanics*, 21, 183-190