Proceedings of the 28th IEEE
EMBS Annual International Conference
New York City, USA, Aug 30-Sept 3, 2006

A system for measurement and calibration of nonorthogonal joints and limbs in humans

Christopher M. Storey, *Member*, *IEEE*, Anne M Hollister, Charles J. Robinson, *Fellow, IEEE,* Norman M. Witriol, Dale O. Anderson, John C. London, and William L. Buford, *Member, IEEE*

***Abstract*—Human limbs are a multilinked system in which the revolute joints are not orthogonal to the limb segments or to each other. The standard method for movements of multilinked systems is the Denavit-Hartenberg (DH) Representation, which is useful for orthogonal systems. When applied to non-orthogonal systems, the DH representation projects the reference frames outside of the limb segments. Computer graphics techniques move arrays of points in bodies that move about arbitrary revolute joints. This computational model has been modified to calculate both position (X, Y, Z) and orientation (yaw, pitch, and roll) of limbs and their individual segments. This method allows a simplified representation for the kinematics of animal limbs.**

I. Introduction

Many mechanical systems are composed of rigid segments linked by simple kinematic mechanisms such as revolute joints. Knowledge of positions and orientations of the links and revolute joints is essential for machine control. The current standard for calculating the position and orientation for the reference frames of these linked mechanisms is the Denavit-Hartenberg (DH) Representation [1]. The DH Representation assumes rigid links between limbs but to simplifies from the defined position and orientation with displacement and Euler angles to four numbers requires orthogonality between limbs and links [1]. The DH is a relatively simple system that works well when the mechanisms are orthogonal. The simplification is the result of choosing coordinate frames for the links that do not have to lie within link or limb [1]. Humans and animals have revolute joints that are not usually perpendicular to each other or to the limb segments, and do not parallel a global reference frame coordinate [2]. Such joints are known as arbitrary revolute joints and can be described as twist and crank angles, a vector from the origin, and the degrees of rotation (pitch – θ) about the revolute joint. DH representation of non-orthogonal systems project the local reference frames outside of the limb segments. Our proposed representation places the limb segment and joint reference frames within the segments or joints, facilitating the measurement, design, modeling, movements and control of these systems via methods developed for computer animation [3]. The parameters, which must be measured in the animal limb, are explicitly stated and are variables in the generalized equations.

Manuscript received April 24, 2006.

C. M. Storey is with the Biomedical Engineering Department at Louisiana Tech University, Ruston, LA 71272 USA (phone: 504-782-3280; fax: 318-213-8856; e-mail: cms043@latech.edu).

A. M. Hollister is with Louisiana State University Health Sciences Center, Shreveport, LA, 71103 USA (e-mail: annahans@mac.com).

C. J. Robinson is with the Center for Rehabilitation Engineering, Science, and Technology, Clarkson University, Potsdam, NY 13699 USA and The Syracuse VAMC Research Service, Syracuse, NY 13210 (e-mail: c.robinson@ieee.org).

N. M. Witriol is with the Physics Department at Louisiana Tech University, Ruston, LA 71272 USA (e-mail: witriol@latech.edu).

D. O. Anderson is with the Mechanical Engineering Department at Louisiana Tech University, Ruston, LA 71272 USA (e-mail: andersonfam@cox-internet.com).

J. C. London is with the Mathematics Department at Tulane University, New Orleans, LA 70118 USA (e-mail: jlondon@tulane.edu).

W. L. Buford, Jr. is with the Research Division of the Department of Orthopaedic Surgery and Rehabilitation at The University of Texas Medical Branch, Galveston, TX, 77555 USA (e-mail: wbuford@utmb.edu)

II. Methods

A. Software

The program was written in Matlab® to have access of the matrix mathematics functions. The program was object-oriented to allow for ease and robustness of expansion. The software provided text (Excel Spreadsheet), jpg (picture), and avi (Video) output representing motion.

B. System representation

The simplest system consisted of two segments and a single joint. The reference frame for the first link was placed at the origin of a Cartesian coordinate system. The displacement vector (D_1), the x, y, and z measurements from the first segment's reference frame to the center of the first revolute, were recorded. The limb local coordinates may be placed anywhere, including a location along the revolute axis. In biological systems, the center of mass in the stationary limb segment is difficult to establish and it changes as the muscles lengthen and contract. The φ_1 and ψ_1 angles were the twist and crank angles of offset from the preceding segment's reference frame needed to align the revolute axis of motion with the z-axis of the preceding limb. In relation to the second limb, the φ_2, ψ_2, and D_2 were the variables measured that were needed to rotate the axis of rotation so as to align with z-axis of the next segment and to find the distance from the joint center to the following segment's center. There for the method provides a definition of the orientation of limbs relative to their joint.

C. Definitions:

A vertices matrix ([$V_{\#}$], Eq. (1)) was defined by the joint number, "#"; therefore, all limbs distal to it were are

operated on or accessed in the matrix. The subscript *d* is equal to the number of the most distal limb. Each limb requires 12 columns of the matrix (expandable for more vertices). The first eight columns were the limb vertices. The ninth column was the limb center, and the 10th, 11th, and 12th columns were the local x, y, and z vectors (not unit vectors) of the local coordinate system relative to global Cartesian coordinate system.

$$\begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{\#,2} \dots & x_{d,12} \\ y_{1,1} & y_{1,2} & \dots & y_{\#,2} \dots & y_{d,12} \\ z_{1,1} & z_{1,2} & \dots & z_{\#,2} \dots & z_{d,12} \\ 1 & 1 & 1 & 1\ 1 & 1 \end{bmatrix} \quad (1)$$

A translation matrix ($[T_{\#,l}]$, Eq. (2)) was defined by a subscript # and *l*, which defined the offset as "from proximal" (fp) or "to distal" (td) to the limb.

$$\begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

Rotation matrices ($[R_{\#,a,l}]$) were defined by #, *l*, and *a*, which was the axis of rotation. Equations (3)-(5) show the rotation matrices for rotating about the *x*, *y*, and *z*-axes respectively. Roll (ψ) was defined by the rotation about the *x*-axis. Yaw (φ) provided the rotation about *y*-axis. Pitch (θ) described the rotation about the *z*-axis.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) & 0 \\ 0 & \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$$\begin{bmatrix} \cos(\varphi) & 0 & \sin(\varphi) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

A joint rotation matrix ([Rev]) defined the amount of rotation about the arbitrary axis of the revolute joint. The ψ and φ angles of offset are calculated from the *z*-axis to allow for the define rotation of the joint to be about the *z*-axis as in Eq. (5). A single rotation matrix ([RM (row, column)]) was used to define the orientation of a limb in space through a single rotation once the yaw, pitch, and roll was tabulated.

$$\begin{bmatrix} c(\varphi)c(\theta) & s(\theta) & -c(\theta)s(\varphi) & 0 \\ -c(\psi)s(\theta)c(\varphi)+s(\psi)s(\varphi) & c(\psi)c(\theta) & c(\psi)s(\varphi)s(\theta)+s(\psi)c(\varphi) & 0 \\ c(\varphi)s(\theta)s(\psi)+s(\varphi)c(\psi) & -c(\theta)s(\psi) & -s(\psi)s(\varphi)s(\theta)+c(\theta)c(\varphi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

In Equation (6), *c* refers to cosine and *s* to sine.

D. System Description

To ease the difficulty in setting up the model with global positions and angles, a relative system was used through an object-oriented design utilizing limb, joint, and system objects. The global origin and axis were designated to be the local origin and axis of the most proximal limb of the system. The limb was modeled as a cuboid, which can easily be changed to any shape to any shaped by adding and/or removing vertices. The cuboid was used to simplify the program and was defined by length, width, and height. The local origin for each limb was defined as its geometric center. The joint was a single axis revolute joint. The center of the revolute joint was at the local origin and aligned with the *z*-axis. The joint was defined in relation to the proximal and distal limbs. Due to the sequential operation in traversing the limb, the program was designed arbitrarily in a proximal to distal method. The measure made for each joint was the x_{fp}, y_{fp}, and z_{fp} offset from the proximal limb. These defined its position in space in relation to proximal limb. Next, the ψ_{fp} (*x*-axis), φ_{fp} (*y*-axis), and θ_{fp} (*z*-axis) was measured as the offset orientation from the local axis of the proximal limb. Then the same offset orientations (ψ_{td}, φ_{td}, and θ_{td}) were measured to rotate the revolute joint to align with the *z*-axis of the distal limb. Finally, the offset was measured from the center of the revolute joint to the geometric center of the distal limb, which provided x_{td}, y_{td}, and z_{td}. No range limitation was enabled to allow coordinates or orientations that are considered out of the range of natural joint motion as in fractures or dislocations.

III. Building the Multi-Linked System

A multi-linked system was assembled to the specifications of an initial state. The local coordinate systems were set at the origin for all limbs. Therefore, the limbs were moved to their appropriate positions and orientations in space via a sequence of matrix multiplications as shown below:

1. $[V_1] = [T_{1,fp}]\ [R_{1,x,fp}]\ [R_{1,y,fp}]\ [R_{1,z,fp}]\ [R_{1,z,td}]\ [R_{1,y,td}]\ [R_{1,x,td}]\ [T_{1,td}]\ [V_1]$
2. $[V_2] = [T_{2,fp}]\ [R_{2,x,fp}]\ [R_{2,y,fp}]\ [R_{2,z,fp}]\ [R_{2,z,td}]\ [R_{2,y,td}]\ [R_{2,x,td}]\ [T_{2,td}]\ [V_2]$
3. . . . . .
4. . . . . .
5. . . . . .
6. $[V_\#] = [T_{\#,fp}]\ [R_{\#,x,fp}]\ [R_{\#,y,fp}]\ [R_{\#,z,fp}]\ [R_{\#,z,td}]\ [R_{\#,y,td}]\ [R_{\#,x,td}]\ [T_{\#,td}]\ [V_\#]$.

IV. Rotation about a Selected Joint

The lack of orthogonality between the joint axes and the global coordinate system presents a problem with joint rotation. To rotate the distal limbs around a respective joint (#), the joint was translated to the global origin and the axis of rotation was aligned with the *z*-axis. The *z*-axis has been designated as the as the only axis of rotation for all limbs. All other axes were held constant at initial specifications as shown in the following sequence:

1. $[V_1] = [R_{1,y,fp}]^{-1}\ [R_{1,x,fp}]^{-1}\ [T_{1,fp}]^{-1}\ [V_1]$
   a. If joint is joint of rotation, jump to step eight.
2. $[V_1] = [T_{1,td}]^{-1}\ [R_{1,x,td}]^{-1}\ [R_{1,y,td}]^{-1}\ [R_{1,z,td}]^{-1}\ [R_{1,z,fp}]^{-1} [V_1]$
3. $[V_2] = [R_{2,y,fp}]^{-1}\ [R_{2,x,fp}]^{-1}\ [T_{2,fp}]^{-1}\ [V_2]$
   a. If joint is joint of rotation, jump to step eight.

4. $[V_2] = [T_{2,td}]^{-1} [R_{2,x,td}]^{-1} [R_{2,y,td}]^{-1} [R_{2,z,td}]^{-1} [R_{2,z,fp}]^{-1} [V_2]$
5. . . . . .
6. . . . . .
7. . . . . .
8. $[V_\#] = [T_{\#,fp}] [R_{\#,x,fp}] [R_{\#,y,fp}] [Rev] [V_\#]$.

After joint was rotated, the system was returned to their proper global coordinates as follows:

1. $[V_\#] = [T_{\#,fp}] [R_{\#,x,fp}] [R_{\#,y,fp}] [R_{\#,z,fp}] [R_{\#,z,td}] [R_{\#,y,td}] [R_{\#,x,td}] [T_{\#,td}] [V_\#]$
2. . . . . .
3. . . . . .
4. . . . . .
5. $[V_2] = [T_{2,fp}] [R_{2,x,fp}] [R_{2,y,fp}] [R_{2,z,fp}] [R_{2,z,td}] [R_{2,y,td}] [R_{2,x,td}] [T_{2,td}] [V_2]$
6. $[V_1] = [T_{1,fp}] [R_{1,x,fp}] [R_{1,y,fp}] [R_{1,z,fp}] [R_{1,z,td}] [R_{1,y,td}] [R_{1,x,td}] [T_{1,td}] [V_1]$.

V. Yaw, Pitch, and Roll of Limbs

The rotation about an arbitrary angle in the multilinked system of limbs allowed distal limbs to have changes in their spatial rotations about more than one axis simultaneously. Therefore, the new yaw, pitch, and roll was calculated after proximal joints have been rotated, but to avoid errors in back calculation of the orientation via position coordinates in real data, the yaw, pitch, and roll were calculated on the fly with rotations through multiplication of only the rotation matrices. Due to the sequence necessary for rotation using Euler angles, the orientations were calculated in the order roll, pitch, and yaw in the back calculation from Cartesian coordinates so the Euler angle multiplication sequence for rotations can take place in the order of yaw, pitch, and roll. The same order was used to calculate in (6). The calculation of yaw, pitch, and roll was tabulated both ways so that each method could verify the other. The method of back calculation was accomplished by first translating the limb back to the global origin of the coordinates system by using the limb center as the offset for the translation matrix (7).

$$\begin{bmatrix} 1 & 0 & 0 & x_{\alpha,9} \\ 0 & 1 & 0 & y_{\alpha,9} \\ 0 & 0 & 1 & z_{\alpha,9} \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{7}$$

The limb number (α) describes the limb to be translated and later rotated. Once the limb was at the origin, the local axis vectors were converted to their respective unit vector coordinates for the y-axis (8)-(10).

$$x\_dc = \frac{x_{\alpha,11}}{\sqrt{x_{\alpha,11}^2 + y_{\alpha,11}^2 + z_{\alpha,11}^2}} \tag{8}$$

$$y\_dc = \frac{y_{\alpha,11}}{\sqrt{x_{\alpha,11}^2 + y_{\alpha,11}^2 + z_{\alpha,11}^2}} \tag{9}$$

$$z\_dc = \frac{z_{\alpha,11}}{\sqrt{x_{\alpha,11}^2 + y_{\alpha,11}^2 + z_{\alpha,11}^2}} \tag{10}$$

Roll (ψ) was calculated by projecting the local *y*-axis unit vector onto *yz*-plane (11) and calculating the angle of rotation (ψ) between *u'* and the global *y*-axis (12).

$$u' = [0, y\_dc, z\_dc] \tag{11}$$

$$\psi = -\frac{\cos^{-1}(u' \cdot [0,1,0])}{|u'|} \times \frac{z\_dc}{|z\_dc|} \tag{12}$$

The arc cosine function only returned values between zero and π radians so it was multiplied by a factor that was -1 or 1 depending on *z_dc* (*z*-component of *y*-axis unit vector) in respect to the *xy*-plane. A similar respective factor was multiplied to determine angle direction for yaw and pitch, also. The computer animation standard rotations utilized for the methodology designate positive angles as counter-clockwise rotations [2]. For ψ, a positive angle required a clockwise rotation to align *u'* with the global *y*-axis. Therefore, the calculated ψ was inverted. The remaining calculations of pitch (θ) and yaw (φ) followed the standard convention of positive angles are for counterclockwise rotations. Using ψ, an *x*-axis rotation matrix was used to rotate the limb so that the local y-axis vector lies in the *xy*-plane.

Since *u"* already lied within the *xy*-plane, but it was calculated with the absolute value of *u'* (13), Pitch (θ) was calculated from the angle of rotation between *u"* and the global *y*-axis (14).

$$u'' = [x\_dc, 0, |u'|] \tag{13}$$

$$\theta = \frac{\cos^{-1}(u'' \cdot [0,1,0])}{|u''|} \times \frac{x\_dc}{|x\_dc|} \tag{14}$$

Yaw (φ) required the use of a separate local axis since the local *y*-axis unit vector was aligned with the global *y*-axis. The local *x*-axis unit vector was calculated (only x and z coordinates; (15) and (16)) to project onto the *xz*-plane (17) and the angle between the *x*-axis vector and the global *x*-axis provides one with the yaw of the limb (18).

$$x\_dc = \frac{x_{\alpha,10}}{\sqrt{x_{\alpha,10}^2 + y_{\alpha,10}^2 + z_{\alpha,10}^2}} \tag{15}$$

$$z\_dc = \frac{z_{\alpha,10}}{\sqrt{x_{\alpha,10}^2 + y_{\alpha,10}^2 + z_{\alpha,10}^2}} \tag{16}$$

$$u''' = [x\_dc, 0, z\_dc] \tag{17}$$

$$\varphi = \frac{\cos^{-1}(u''' \cdot [1,0,0])}{|u'''|} \times \frac{z\_dc}{|z\_dc|} \tag{18}$$

Utilizing the calculated yaw, pitch, and roll, one is able to move and orient the limb without sequential steps as in (19).

$$[V_\#] = [T] [R_\psi] [R_\theta] [R_\varphi] [V_\#] \tag{19}$$

The lack of ability to make small measurements for the position of limbs introduces the possibility for large errors when back calculating the yaw, pitch, and roll for a limb, especially at the asymptotes. So in addition to back calculating the yaw, pitch, and roll from the limb's position relative to global axis, the yaw pitch and roll was calculated

by only using the inputted rotation matrices. The rotation matrices were ordered as they would for multiplication for building of limb as previously shown, but no translations are used. A rotation matrix in (6) is obtained and by using an ordered sequence of rotation matrix multiplications equations are given to back calculate yaw, pitch, and roll from the values in the rotation matrix as follows:

1. $[RM_1] = [R_{1,x,fp}]\,[R_{1,y,fp}]\,[R_{1,z,fp}]\,[R_{1,z,td}]\,[R_{1,y,td}]\,[R_{1,x,td}]$
2. $[RM_2] = [R_{2,x,fp}]\,[R_{2,y,fp}]\,[R_{2,z,fp}]\,[R_{2,z,td}]\,[R_{2,y,td}]\,[R_{2,x,td}]$
3. . . . . .
4. . . . . .
5. . . . . .
6. $[RM_{\#}] = [R_{\#,x,fp}]\,[R_{\#,y,fp}]\,[R_{\#,z,fp}]\,[R_{\#,z,td}]\,[R_{\#,y,td}]\,[R_{\#,x,td}]$

The pitch was calculated first, since it is in the equation from the (6) that has one unknown. The arc-sine function has a range $[-\pi/2, \pi/2]$, but since the order of yaw, pitch, and roll, roll was rotated first this guarantees that the pitch will always be less than 90°.

$$\theta = \sin^{-1}\left(RM(1,2)\right) \tag{20}$$

Roll as in (12) was negated to provide the correct rotation direction. The *z_dc* vector value was the same for roll and yaw as calculated in (10) and (16) respectively. Since the arc cosine's range was $[0, \pi]$, the *z_dc* vector value was used to determine the sign and the direction of the rotation.

$$\psi = -\cos^{-1}\left(\frac{RM(2,2)}{\cos(\theta)}\right) * \left(\frac{z\_dc}{|z\_dc|}\right) \tag{21}$$

$$\varphi = \cos^{-1}\left(\frac{RM(1,1)}{\cos(\theta)}\right) * \left(\frac{z\_dc}{|z\_dc|}\right) \tag{22}$$

This method allows one to track the yaw, pitch, and roll of the limbs without the need for position data (except to track the sign for yaw and roll).

A. Results

To verify our methods we rotated to position and then back to initial position with different rotations on the way back to the home position. Our final vertices matrix equaled the initial to verify our method since a closed loop rotation is an identity matrix. Table 1 shows the sequence of rotations and the initial offsets that are illustrated in Fig. 1. Fig. 1 is the display of a simple system of only two arbitrary revolute joints since it the points were rotated as a matrix of vertices, which makes it simple to expand to detailed objects. The yaw, pitch, and roll was calculated both through rotations and by back calculation from position in both joint with rotations from -180° to 180° in 1° increments and were found equal when compared.

VI. A Model for Calibration of Ankle Angle Measurements

A model of the ankle was developed based on three segments with two arbitrary revolute joints. The segments are the calcaneus, talus, and mortise (comprised of the leg bones and ligaments). The arbitrary revolute joints consist of the talocrural joint and the subtalar joint [4], [5]. This allows us to view the ankle bones as they rotate naturally instead of about the assumed single orthogonal axis of most models [6]. This will help to increase the accuracy of measurements of ankle rotation. The determination of joint moments and reaction forces make it possible for more realistic and natural ankle prostheses.

VII. Conclusion

This computational approach provides calculations for the position and orientation of limbs and revolute joints throughout the system's motion. The method facilitates modeling of kinetics and kinematics of joint in human and animal limbs and further force analysis. The ability to calibrate accurately small rotations of nonorthogonal joints improves the measurement of orientation of limbs. The methodology allows for measurements and calculations of joints without the need for any mutually orthogonal axes that could lie outside the body of rotation.

References

[1] Spong, M. and Vidyasagar, M., *Robot dynamics and control*, 1989, John Wiley & Sons, pp. 62-72.

[2] Koti, N., "Design of a biomimetic manipulator with nonorthogonal Joint Axes," Louisiana Tech University, Thesis, 2005, unpublished.

[3] Hearn, D. and Baker, M. P., *Computer graphics C version*, 1997, Prentice Hall, pp. 408-423.

[4] Isman, R. E. and Inman, V. T., "Anthropometric studies of the human foot and ankle," Bulletin of Prosthetics Research, 1969, 97-129.

[5] Inman, V. T., *The joints of the ankle*. Baltimore: Williams & Wilkins, 1976.

[6] Singh, A. K., Starkweather, K. D., Hollister, A. M., Jatana, S., Lupichuk, A. G., "Kinematics of the ankle: a hinge axis model," *Foot & Ankle* Vol. 13 No. 3 pp. 439-446, October 1992.

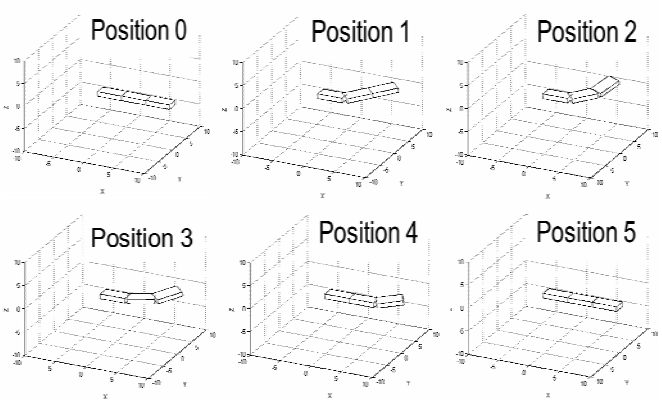

Fig. 1. Displays positions of a two joint system going through a series of rotations defined in Table 1.

TABLE 1
SERIES OF ROTATIONS

Joint Offsets	φ_{fp}	ψ_{fp}	θ_{fp}	φ_{td}	ψ_{td}	θ_{td}
1	10°	5°	0°	-10°	-5°	0°
2	20°	15°	0°	-20°	-15°	0°

Position	Rotated Joint	Degrees
0	None	
1	1	45°
2	2	30°
3	1	-25°
4	1	-20°
5	2	-30°