2008-01-1922

Multiple User Defined End-Effectors with Shared Memory Communication for Posture Prediction

Brent Rochambeau, Timothy Marler, Anith Mathai, and Karim Abdel-Malek
Virtual Soldier Research (VSR) Program, University of Iowa

ABSTRACT

Inverse Kinematics on a human model combined with optimization provides a powerful tool to predict realistic human postures. A human posture prediction tool brings up the need for greater flexibility for the user, as well as efficient computation performance. This paper demonstrates new methods that were developed for the application of digital human simulation as a software package by allowing for any number of user specified end-effectors and increasing communication efficiency for posture prediction. The posture prediction package for the digital human, Santos™, uses optimization constrained by end-effectors on the body with targets in the environment, along with variable cost functions that are minimized, to solve for all joint angles in a human body. This results in realistic human postures which can be used to create optimal designs for things that humans can physically interact with. Previously the end-effectors could only be specified in relation to the left and right wrist and ankle joints. Since the tool was still in developmental phases, communication between the software used to visualize the digital human and environment was done through file I/O. A new optimization method has been developed and implemented to allow for any number of user specified end-effectors, which can be in relation to any joint in the body. Each end-effector can be constrained to any individual target in the environment, which allows for much more flexible interface for a user to define the boundaries of predicting human posture. Communication speeds were increased on average by almost six times through the use of creating a shared memory block, which can be accessed by posture prediction code and the application to visualize the resulting postures at the same time. The combined results of these additional features for posture prediction allow for dynamically updating and visualizing of posture prediction results as new targets for any part of the human body are created or changed in the environment. This in turn provides a new intuitive method for creating a posture prediction simulation which is more interactive with the user.

INTRODUCTION

A key element with human modeling and thus with product design is posture prediction. Posture Prediction gives the ability to predict human posture accurately and quickly and then provides posture-related feedback to the user. In response to this need, The Virtual Soldier Research (VSR) Program has developed an optimization-based approach to posture prediction that operates in real time and includes a variety of features. This paper presents two new advances with optimization-based posture prediction. First, a method for specifying end-effectors anywhere on the body was developed. Secondly, a technique was utilized for leveraging shared memory as a method of communication between the posture prediction code and the user interface, which substantially reduces the total communication time. Combining the two advancements allows a user to manipulate any point on an avatar in real time, while all other components of the avatar are simultaneously governed by posture prediction. The result is a powerful new product for human-centric design.

Predicting human posture is often limited by the location of *end-effectors*. End-effectors are points of interest on the avatar and are typically constrained to specified locations in the avatar's environment. When end-effectors are only related to the end of link segments, such as the wrists and ankles, the user is limited to conducting posture prediction studies with which the avatar touches targets with only the hands or feet. As computer aided product design become more popular in an effort to reduce physical prototypes, and thus save money and time, there is a growing need for more advanced posture prediction simulations that go beyond using points on the hands and feet as end-effectors.

Since posture prediction is the key feature that the application described in this paper advances, a brief overview of different posture prediction approaches and advances is provided first. Currently, there are two fundamental approaches to predicting postures, on the research side as well with commercially available tools. One method predicts what human posture looks like based on prerecorded motion capture, anthropometric

data, and functional regression models (Beck and Chaffin, 1992; Zhang and Chaffin, 1996; Faraway, 1997; Das and Behara, 1998; Faraway et al, 1999; Chaffin, 2002).

The second method of predicting posture is done with traditional inverse kinematics, which does not use any observed data. A common approach to inverse kinematics is called the pseudo-inverse method. With such methods, the motion of each link segment is modeled to formulate a set of governing joint equations (Jung et al, 1995; Jung and Choe, 1996; Wang, 1999; Tolani and Badler, 2000). With this method however, as the model's degrees of freedom increase, the systems of equations become increasingly challenging to solve.

A different inverse kinematics method involves optimization. Optimization is used to find a set of joint angle values (each subject to their own constraints), that are used to minimize certain human performance measures, such as discomfort. The constraint in the optimization problem is restricting an end-effector to reach a target point (Abdel-Malek et al, 2001; Mi et al, 2002). This approach does not require any prerecorded data and can be computed efficiently (Farrell and Marler, 2004). Most recent advancements concerning the number of end-effectors in the posture prediction problem have been creating a dual-arm posture prediction, where end-effectors are placed at the end of each arm, each having their own individual constraint (Farrell and Marler, 2005; Yang et al 2004, 2006, 2007). When legs are included in the model, end-effector can also be placed at the end of each leg, since it is also the end of each link segment. The development of a multiple end-effectors feature described in this paper extends the current state of the art and allows a user to place any number of end-effectors on any part of the body. This is a new capability with respect both to research efforts and currently available human-modeling products. In addition to incorporating multiple end-effectors, *shared memory* is leveraged to increase speed. Shared memory allows multiple programs to access the same memory location, so when one program changes a variable's value, all other programs that are allowed immediately have access to that change. This makes it an efficient method of passing data. If posture prediction is to provide an efficient design tool, computational efficiency is critical. Trade-off analysis of various design alterations necessitates the use of real-time simulations, and shared memory can enable this. Shared memory is typically used as a method of inter-process communication, which is a way of exchanging data between multiple programs running at the same time. When applications have a need for more efficient communications and higher performance, parallel processing is considered, because multiple processes can run simultaneously. Using shared memory for this function allows one to avoid expensive overheads associated with other parallel methods (Wang et al, 1999). However, shared memory has never been applied to posture prediction as a way of increasing speed. To increase the speed of posture prediction to the point where a user can drag any point on the body and the simulation can concurrently predict the consequent joint angles in real time provides a new tool that saves time and money during ergonomic design studies.

The following section in this paper provides an overview of optimization-based posture prediction and how it is integrated with 3D visualization. We then discuss the advantages of, and a method for allowing the user to specify multiple end-effectors anywhere on the body. After that, shared memory is addressed and how it is implemented with posture prediction. Finally, we show how combing multiple end-effectors and shared memory allows for a new intuitive tool for human modeling product design, which allows the user to drag any part of the body with realistic human postures being predicted and visualized in real time.

OPTIMIZATION-BASED POSTURE PREDICTION

This section discusses the fundamental of optimization-based posture prediction, as implemented in Santos™, a new kind of virtual human developed at VSR (Abdel-Malek et al, 2006). Marler et al (2005) explain how the most basic posture prediction problem entails having an avatar use a natural posture to contact a specified target point with an end-effector on a kinematic system such as a human arm. With an optimization-based approach, the joint angles for all of the degrees-of-freedom (DOFs) in the human model provide the design variables and are determined by optimizing an objective function that represents a human performance measure. These performance measures can include discomfort, joint displacement, potential energy, effort, visual acuity, etc.

The interface and all interaction with the user are done through the Virtools 3D development engine. To date, this interaction has been conducted as follows. The user can select through posture prediction options, such as the performance measures mentioned above, toggle vision on or off, and toggle collision avoidance on or off. There are end-effectors in relation to the end of each link series (left and right wrists and ankles), which the user can specify targets in the world for each (Farrell et al, 2005). Every time a new target is selected for one of these end-effectors, Virtools writes out the information posture prediction needs to input files and runs posture prediction as an executable. When posture prediction is run, it reads the input files, calculates the optimal solution of joint angles, and then writes out that solution to a separate output file. Virtools, which has been waiting during this time, then reads the solution file and applies calculated joint angles to the visualized avatar. This entire process can be seen in Figure 1, where each file represents different sets of input data for posture prediction.

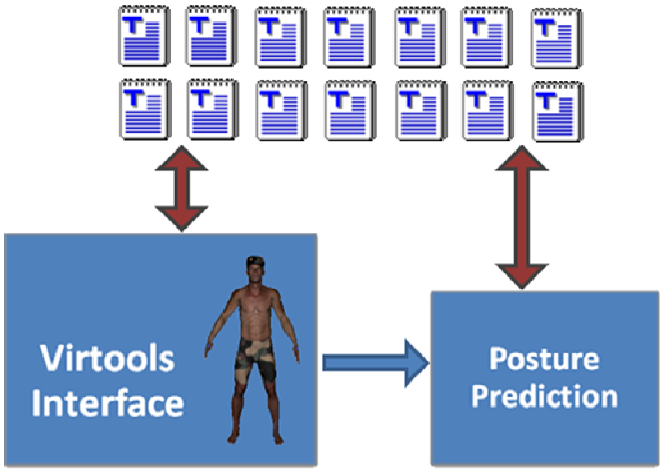

Figure 1. Previous method of communication between Virtools and posture prediction: file I/O with multiple files containing different sets of data.

The use of file I/O as a method of communication was originally chosen for developmental purposes and ease of debugging. Overall, the whole posture prediction process takes about .05 seconds, depending on the complexity of the optimization problem. This speed is fast enough for a real time prediction, but not fast enough for a user to drag postures with acceptable results.

MULTIPLE END-EFFECTORS

The addition of the multiple end-effectors feature to posture prediction gives the user more freedom and flexibility to setup a posture prediction simulation. Previously, the posture prediction code could only calculate postures for a set number of end-effectors that also could only be specified in relation to the end of segment links (left and right wrists and ankles). A picture of the upper body end-effector locations can be seen in Figure 2.

Figure 2. Previously, upper body end-effectors could be specified in relation to the left and right wrist joints.

In order to give the user complete freedom to set up a posture prediction simulation, he or she should be allowed to select any part of the digital human's body as an end-effector, and give that point an individual target in the world. This is the basic problem that the multiple end-effectors feature addresses. In order to accomplish this, the posture prediction code needs to be able to perform inverse kinematics for segments in the human model other than the link ends, and also accept any number of end-effector/target pairs as input. Once that is complete, the user interface needs to be updated to communicate these new inputs to the posture prediction code, as well as updated to allow a user to select any part of the digital human body.

Since there can now be any number of end-effectors accepted by the code, the first important change was that the variables related to end-effectors and their targets must now by dynamic. Whenever the code is executed it should now dynamically allocate the memory for the necessary arrays associated with those variables. The optimization problem set up in the code must also be changed dynamically since each end-effector and target are individual constraints that must be satisfied.

Posture prediction with Santos™ uses the Denavit-Hatenberg method (Denavit and Hartenberg, 1955). With this method, there are individual transformation matrices that describe the relationship between connected joints in a robotic model. In the Figure 3, the first spine joint would be the bottom right degrees of freedom, which acts as the global position of the world for Santos™, and the right wrist would be the top left degrees of freedom. Targets for the end-effectors are given in the world with respect to that global spine joint, so it is necessary to know how far away an end-effector (which are always defined by the local coordinates of the parent joint) are from the global coordinate system.

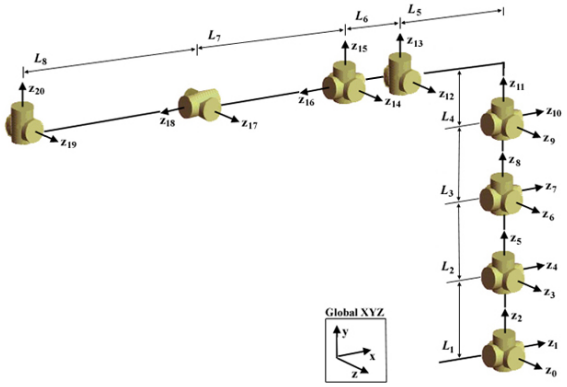

Figure 3. Each joint has its own set of local coordinate systems and there is a transformation matrix that describes how to get from one to the next.

Previously, to compute the location of an end-effector in relation to a global root coordinate system, the posture

prediction code would start at the first spine joint, and then multiply each transformation matrix along the link chain out to the end-effector related to the wrist. Once all transformation matrices are multiplied out and then multiplied by the local vector related to the last coordinate system, the global position of the end-effector is known. This can then be compared to the global position of its target. The distance between the two is then used in the optimization problem to be constrained to zero. With the inclusion of a dual-arm system, the code would have to compute this twice, one for each wrist end-effector (Farrell et al, 2005).

In order to use multiple end-effectors, the code must now loop for each end-effector. End-effectors are related to their closest parent joint (i.e. end-effector on the forearm is related to the last elbow joint). However, since end-effectors can now be placed anywhere on the body, the multiplication of transformation matrices for a certain end-effector must stop at that parent joint, and then compute the position from the global root joint. Figure 4 demonstrates the relationship between end-effectors on the human body and the parent coordinate system they are taken with respect to. The distance between each end-effector and target can now be computed, and the optimization code can run until each pair or end-effectors and targets is essentially equal, which results in a realistic human posture for the digital human while end-effectors are constrained to touching the targets. This solution is expressed in joint angles for the entire body.

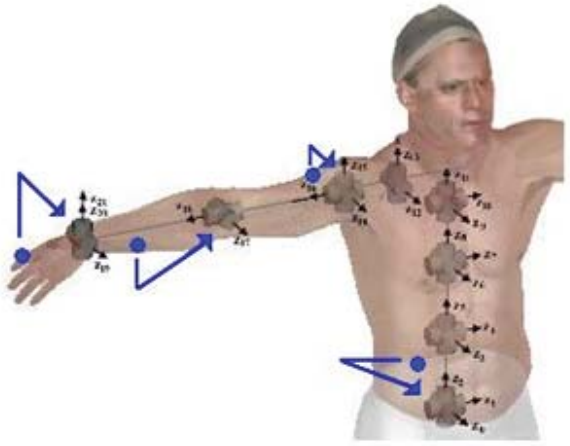

Figure 4. Several end-effectors (blue spheres) and their local parent joint coordinate systems in which their coordinates are related to.

On the interface side, end-effectors are also now variable and dynamic. Every time a user creates a new end-effector on the digital human's body, its local position in relation to its parent joint must be determined. End-effector local positions and the targets associated to each end-effector are inputs to the posture prediction code. The interface must make sure to only execute posture prediction when a new end-effector and target pair has been created by the user (only creating new targets without paired end-effectors or vice versa should not result in a new posture).

Combing the interface and code together now allows the user to setup a posture prediction simulation with point constraints anywhere on the body. An example of this can be seen in Figure 5 with blue end-effector spheres and white target spheres.

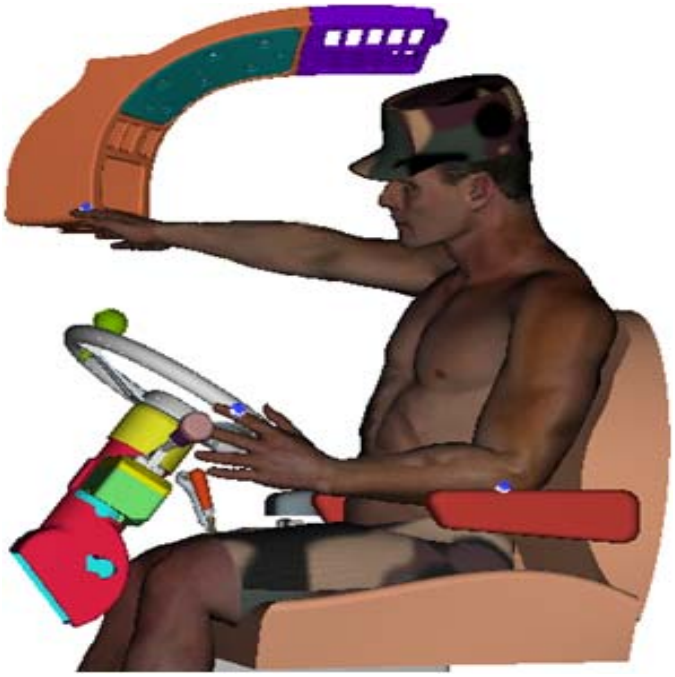

Figure 5. Predicted posture with end-effectors (blue spheres) placed on the multiple locations on the digital human's body, each given its own target (white spheres).

SHARED MEMORY

Given the ability to specify any number of targets and associated end-effectors, we now develop the ability to communicate faster and simultaneously perform posture prediction. That is, posture prediction operates in real time as various points on the avatar are positioned. Whereas currently available animation and surface modeling tools (Maya, 3D Studio Max, etc.) have basic inverse kinematics capabilities that essentially yield random postures for a specified target point, Santos^{TM} incorporates what we call *Advanced IK* to predict realistic human postures.

However, constantly opening/closing and reading/writing of files (Figure 1) adds unnecessary overhead and delays, so a more sophisticated communication method was needed. Shared memory is a solution that allows both processes to have access to the same memory block of common variables. When one process changes a variable value, the other process immediately has access to the new value. This results in much faster communication and synchronization between multiple processes.

As mentioned previously and shown in Figure 1, the following process is used for communication between Virtools and the posture prediction code (the previous methods that get improved through shared memory implementation are colored in red):

1. Virtools opens files, writes information, and closes the files
2. Posture prediction execution begins
3. Posture prediction opens files, reads information, and closes the files
4. Posture prediction calculates optimal human posture solution
5. Posture prediction opens files, writes the solution, and closes the files
6. Virtools opens files, reads the solution, and closes the files
7. Virtools applies the solution visually to the avatar

Shared memory replaces this slow and primitive method of communication. Shared memory is achieved by creating a shared memory DLL. A DLL stands for dynamic-link library and can include programmed subroutines, data, and resources that can be used by multiple platforms and programming languages. In a shared memory DLL, all of the variables that are shared among multiple processes are defined, and then get and set functions for each of these variables are also defined. To have access to these variables, each specific process needs to load the shared memory DLL and use the get and set functions to either retrieve a value or change it. A basic diagram of two processes using a shared memory DLL can be seen in Figure 6.

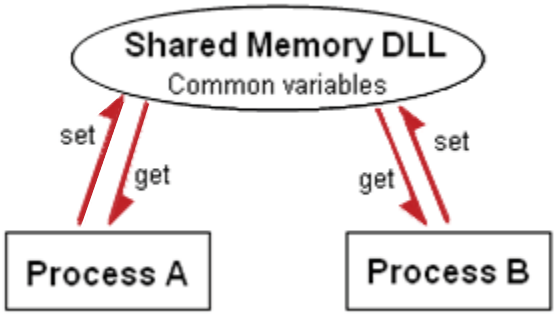

Figure 6. Basic shared memory diagram with multiple processes able to get and set common shared variables defined in the shard memory DLL.

In order to use shared memory, the communication sections of the posture prediction code need to be changed. Before any communication occurs though, the shared memory DLL needs to be loaded into the program. Once it is loaded the code has access to all the get and set functions, which in turn gives it access to all the shared variables. The posture prediction code is changed by replacing all the *read* statements with *get-variable* function calls, and all the *write* statements are replaced with *set-variable* function calls.

The Virtools interface was changed by removing all the *building blocks* that write out the posture information. A building block is a visual representation of a block of code that achieves a specific function, which is used in Virtools interface development. Once the write file building blocks were removed, a new building block was created. This new building block first loads the shared memory DLL, and then sets all of the appropriate values before posture prediction is executed. Once posture prediction computes an optimal posture and the values have been set back to the shared memory location, Virtools can then access them and apply the results. Overall, the new process when a user wants a new posture predicted can be described as follows (the improved methods are colored in red):

1. Virtools loads the shared memory DLL and sets all posture values
2. Posture prediction execution begins
3. Posture prediction loads the shared memory DLL and gets all the input values
4. Posture prediction calculates optimal human posture solution
5. Posture prediction opens sets the solution in shared memory
6. Virtools gets the solution
7. Virtools applies the solution visually to the avatar

The steps are now simplified, and each one does not take as long as they previously did using file I/O.

One particular challenge that was overcome was the implementation of shared memory DLLs with Fortran code. Since the posture prediction code is written in Fortran, it can only load DLLs that are in the same directory as the actual executable. Since the shared memory DLL may not be in that directory, a method needed to be made to specify the path of the DLL. Since the C++ language is capable of loading remote DLLs, a C++ wrapper DLL was created. The posture prediction code can load this local wrapper DLL, which can then load the remote shared memory DLL. This entire process can be seen in Figure 7.

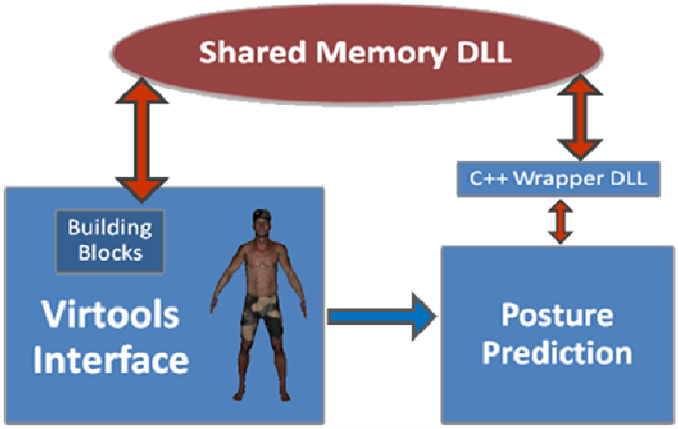

Figure 7. Posture prediction shared memory process with Santos™

While the C++ wrapper does add a little more overhead, the shared memory communication process is still significantly faster than file I/O. Averaging 100 trials for each under the same conditions, the communication speed with file I/O came out to be .024 seconds while shared memory was .0041 seconds. This is 5.85 times faster than the previous method of communication, which is a valuable improvement that was needed in order for the tool described in this paper to be possible. Now, Instead of the entire posture prediction process taking a split second from a user's perspective, it now appears to be instantaneous with the posture prediction code and user interface synchronized.

COMBINING MULTIPLE END-EFFECTORS AND SHARED MEMORY

With posture prediction running at this faster speed, it meets the speed requirement to be called constantly in a loop while Virtools is also constantly looping and updating the avatar with the predicted posture. A user can click and drag targets around just as fast as the posture to touch that target is computed, so there appears to be motion. Combining this with multiple end-effectors posture prediction gives the user the ability to click and drag any part of the body, but while dynamically displaying realistically predicted postures. This is a new and intuitive method to allow users to setup or change posture prediction simulations.

Some changes needed to be made to both the flow of logic in the posture prediction code as well as the interface to achieve this new feature. Once a user clicks a point somewhere on the avatar's body (creating an end-effector), both the posture prediction code and user interface enter a mode where they are running at the same time and continually communicating the input and results with each other. This mode then stays on as the mouse is down. Now as the mouse is dragged, the target for that individual end-effector is updated to that mouse point, which posture prediction uses as the target input. The visualization of the joint posture for the avatar is updated dynamically. As seen in Figure 8, the blue sphere represents both the end-effector and the target as it is dragged smoothly in one motion.

Figure 8. End-effector being applied to the right hand and then dragged as postures are updated.

Since the posture prediction code has multiple end-effectors capabilities, any part of the body can be chosen as an end-effector and then dragged. As new end-effectors are added, the previous end-effectors stay constrained to their given target. Users can also easily delete any end-effector or target, or update a current value. Currently the interface is set up so that a left click creates end-effectors, and a right click can drag any current point already defined, or replace the closest end-effector with the current click. Since it is still in early stages, there are still many possibilities for further development.

CONCLUSION

Throughout this paper a major evolution of the Santos™ posture prediction code and user interface have been described. This has resulted in a new intuitive method for running posture prediction for digital humans. This was accomplished through a new method for using any number of user-specified end-effectors with optimization-based posture prediction. In addition, a new methodology for incorporating shared memory was leveraged to provide a more efficient way for multiple processes to communicate with each other. This increased computational speed substantially. Combining the multiple end-effectors capability with shared memory communication results in a novel tool for human posture analysis and product design. This represents not only a practical and intuitive product but also the significant advancement in the field of human posture prediction.

Typically, posture analysis is conducted by having an expert user laboriously position and orient and digital mannequin. This is the case with many other human modeling software packages since they do not have realistic human postures that are computed. Positioning a digital mannequin into a posture that looks realistic may not only produce inaccurate results, but this can take a significant amount of time as well. Engineers or ergonomists are needed to analyze the resulting posture and recommend design changes. The capabilities presented in this paper promise to revolutionize this process, and save time and money since a designer will be able to interact with a human posture prediction tool dynamically while being able to drag around any part of the avatar's body. Santos's™ new posture-prediction functionality provided a much needed alternative to currently available inverse kinematics tools.

Future work for the posture prediction code will involve a complete conversion to C++, which will thus eliminate the need for a shared memory wrapper for communication, and also increase the efficiency. In addition, using shared memory as a faster method of communication can be applied to other features of human modeling software products as well, which will increase the communication efficiency of a whole human modeling software package. Since this was the first iteration of this Advanced IK tool which was made to work with an already existing code, future iterations with

code redesigns may prove to have more improvements in structure and performance.

ACKNOWLEDGMENTS

This research was funded by the US Army TACOM project: Digital Humans and Virtual Reality for Future Combat Systems (FCS), and the Caterpillar project: Digital Human Modeling and Simulation for Safety and Serviceability.

REFERENCES

1. Abdel-Malek, K., Yang, J., Marler, T., Beck, S., Mathai, A., Zhou, X., Patrick, A., and Arora, J. (2006), “Towards a New Generation of Virtual Humans,” *International Journal of Human Factors Modeling and Simulation*, 1 (1), 2-39.
2. Abdel-Malek, K., Yu, W., and Jaber, M., (2001), “Realistic Posture Prediction,” 2001 SAE Digital Human Modeling and Simulation.
3. Beck, D. J., and Chaffin, D. B. (1992), “An Evaluation of Inverse Kinematics Models for Posture Prediction”, Computer Applications in Ergonomics, Occupational Safety and Health, Elsevier, Amsterdam, The Netherlands, 329-336.
4. Chaffin, D. B. (2002), “On Simulating Human Reach Motions for Ergonomic Analysis”, Human Factors and Ergonomics in Manufacturing, 12, (3), 235-247.
5. Das, B., and Behara, D. N. (1998), “Three-Dimensional Workspace for Industrial Workstations”, Human Factors, 40, (4), 633-646.
6. Denavit, J. and Hartenberg, R.S., (1955), “A kinematic notation for lower-pair mechanisms based on matrices", Journal of Applied Mechanics, Vol. 77, pp. 215-221.
7. Faraway, J. J. (1997), Regression Analysis for a Functional Response”, Techometrics, 39, (3), 254-262.
8. Faraway, J. J., Zhang, X. D., and Chaffin, D. B. (1999), “Rectifying Postures Reconstructed from Joint Angles to Meet Constraints”, Journal of Biomechanics, 32, 733-736.
9. Farrell, K. and Marler, R.T., (2004), "Optimization-Based Kinematic Models for Human Posture", University of Iowa, Virtual Soldier Research Program, Technical Report Number VSR-04.11.
10. Farrell, K., Marler, R.T., and Abdel-Malek, K., (2005) “Modeling Dual-Arm Coordination for Posture: An Optimization-Based Approach”, University of Iowa, Virtual Soldier Research Program, SAE paper 2005-01-2686.
11. Jung, E.S. and Choe, J., (1996), “Human reach posture prediction based on psychophysical discomfort”, International Journal of Industrial Ergonomics, Vol. 18, pp. 173-179.
12. Jung, E.S., Kee, D., and Chung, M.K., (1995), “Upper body reach posture prediction for ergonomic evaluation models”, International Journal of Industrial Ergonomics, Vol. 16, pp. 95-107.
13. Marler, R.T., Rahmatalla, S, Shanahan, M, and Abdel-Malek, K., (2005) “A New Discomfort Function for Optimization-Based Posture Prediction”, University of Iowa, Virtual Soldier Research Program, SAE paper 2005-01-2680.
14. Mi, Z., Yang, J., and Abdel-Malek, K., (2002), "Real-Time Inverse Kinematics for Humans," Proceedings of 2002 ASME Design Engineering Technical Conferences, DETC2002/MECH-34239, September 29-October 2, Montreal, Canada.
15. Tolani, D., Goswami, A., and Badler, N., (2000), “Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs”, Graphical Models, Vol. 62, No. 5, pp. 353-388.
16. Wang, X.G., (1999), “A behavior-based inverse kinematics algorithm to predict arm prehension postures for computer-aided ergonomic evaluation”, Journal of Biomechanics, Vol. 32, pp. 453-460.
17. Wang, Shi-You, Guo, Fu-Shun, Shuo-Ben, Bi, and Zang, Tian-Yi, (1999), “An improved inter-process communication mechanism using shared memory”, Mini-Micro Systems, v 20.
18. Yang, J., Marler, R.T., Kim, H., Arora, J., and Abdel-Malek, K., (2004), “Multi-Objective Optimization for Upper Body Posture Prediction,” 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Aug. 30-Sept. 1, 2004, Albany, New York, USA.
19. Yang, J., Marler, R.T., Beck, S., Abdel-Malek, K, and Kim, J., (2006) “Real-Time Optimal Reach-Posture Prediction in a New Interactive Virtual Environment,” Journal of Computer Science and Technology, Vol. 21, No. 2, 2006, pp. 189-198.
20. Yang, J., Kim, J., Abdel-Malek, K., Marler, T., Beck, S., and Kopp, G., (2007) “A New Digital Human Environment and Assessment of Vehicle Interior Design,” Computer-Aided Design, Vol. 39, 2007, 548-558.
21. Zhang, X., and Chaffin, D. B. (1996), “Task Effects on Three-Dimensional Dynamic Postures During Seated Reaching Movements: An Analysis Method and Illustration”, Proceedings of the 1996 40th Annual Meeting of the Human Factors and Ergonomics Society, Philadelphia, PA, Part 1, 1, 594-598.