Adapting Motion Capture Data Using Weighted Real-Time Inverse Kinematics

MICHAEL MEREDITH AND STEVE MADDOCK
University of Sheffield, Sheffield, U.K.

In this article we present a technique that enhances an inverse kinematics (IK) solver such that when the results are applied to a computer character, we can generate a level of individualization tailored to both the character and the environment, e.g., a walking motion can become "stiffer" or can be turned into a limping motion. Since the technique is based on an IK solver, we also have the desirable effect of solving retargetting issues when mapping motion data between characters. As the individualization aspect of our technique is very tightly coupled with the inverse kinematics solver, we can achieve both the individualization and retargetting of characters in real time.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism--*Animation*

General Terms: Design, Experimentation, Performance

Additional Key Words and Phrases: Software, games, animation, inverse kinematics, motion capture data

1. INTRODUCTION

One of the main issues in using motion capture data to animate human figures is the problem of retargetting. This is the issue of applying motion data captured from one person to a virtual person of a different size. Since such data is usually in the form of joint angles relating a hierarchy of pieces, a person with different limb lengths driven by the same angles will have a different end-limb point. This is perhaps most clearly illustrated when characters' feet appear to slide across the floor or penetrate it. A simple solution would be to capture data from a real person of the same size as the virtual character. However, this is inflexible, and would not work for some game characters, e.g., monsters. Thus we need a more flexible approach. This is offered by Inverse Kinematics (see Watt and Watt [1992] for a general introduction to IK), which can be used to adjust a motion. The use of inverse kinematics in the field of character animation is not a new idea; however, it is only recently being exploited in real-time applications such as games.

In additional to the retargetting properties that come with an IK solver, we have further enhanced the use of our IK technique to incorporate a level of stylization control over the character that comes at no extra computation cost. We can transform a single base motion to a character of different physiological build, and we can further adapt the motion in real time to simulate the appearance of injuries. Thus a character in a game could receive an injury and change his motion accordingly in real time, using our techniques for adapting existing normal motion.

Authors'addresses: Department of Computer Science, University of Sheffield, United Kingdom; email: {M.Meredith, S.Maddock}@dcs.shef.ac.uk

The following section gives a review of current approaches to adapting character motion. Then, after presenting the details of our technique, we demonstrate the principle by applying it to a single walking motion that we adapt to demonstrate both individualization and injury simulation.

2. RELATED WORK

There are two general approaches to acquiring base motions for character animations. One way of obtaining the data is to record the motion of a live subject using motion capture technology (mocap) [Vicon Motion Systems Ltd. 2004], while the other technique requires the motion to be simulated. The latter can itself be further broken down into several different techniques, including keyframing, inverse kinematics [Tolani et al. 2000; Zhao and Badler 1994] and dynamics [Brogan et al. 1998] algorithms.

In many cases, it is desirable to adjust motions to meet specific environmental constraints or properties of the computer character. Generally speaking, adaptations are added onto the base motions or variations of the simulation algorithm are used, rather than creating completely new algorithms to generate such changes. One of the first changes that generally need to be done is to retarget the motion to a character that may have different dimensions. This is done to eliminate visual artefacts that result from motion mapping. It has been successfully tackled in the past with a variety of different techniques, including IK [Fedor 2003.], spacetime constraints [Gleicher 1998], and dynamics [Hodgins and Pollard 1997]. The former techniques tend to be less computationally expensive than the latter ones, and in particular, the IK algorithm we use in this article has the ability to perform real-time retargeting, in addition to the individualization we present.

Beyond the task of retargetting characters lies the field of adapting motions to portray more complex stylization attributes such as physiological build (individualization) and emotion. In the past, much of the work in introducing different physiological builds into character motions was based on dynamics and biomechanics [Hodgins et al. 1995; Komura and Shinagawa 2001]. These techniques demonstrate good realism in the results, but at a large computational cost that would not be available in real-time applications. This is where our technique demonstrates its potential.

Another area of interest for adding stylization to base motions is in simulating a level of visible emotion. Various techniques have been used to achieve this goal, including the use of Fourier principles [Kraus 2004; Unuma and Takeuchi 1993; Unuma et al. 1995], energy consumption [Park et al. 1997], and emotional state [Densley et al.1997]. The latter technique introduces emotion to the character's appearance by constraining joint angles based on the emotional state of the character. Although we do not investigate this here, it would be possible to incorporate it into our work, to enhance the individualization we will demonstrate later in this article, adding further value to our design.

In the following section we describe the technique that allows us to adapt an existing motion-captured animation that is retargeted to both the environment and the character and to add extra richness to the motion in the form of individualization, including injuries. All of this is achieved in real time, unlike many of the existing techniques that rely on complex dynamics to achieve the same aim.

3. CHARACTER INDIVIDUALIZATION

Our system, *MovingIK SE*, is comprised of three independent modules that communicate using a level of parameterisation that allows flexible control over the generated motions. The system's modular design is outlined in Figure 1.

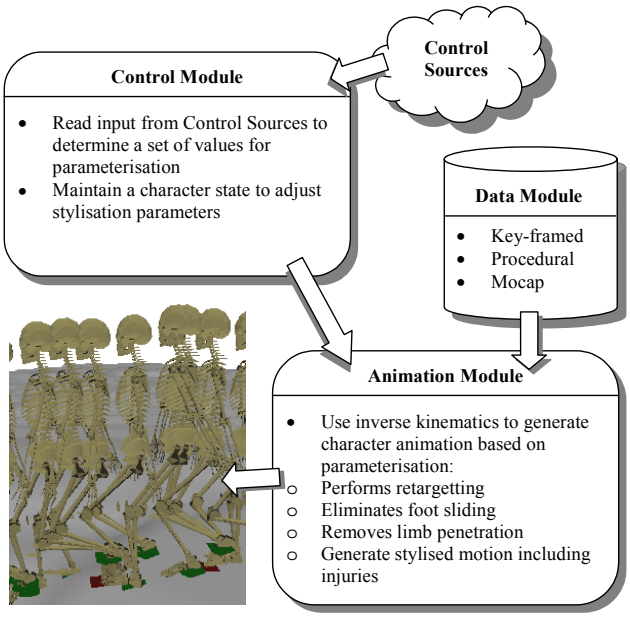


Fig. 1. Control structure of *MovingIK SE*.

The control module is the top-level component whose purpose is to generate a set of values for the parameterized motion. The control module determines the parameterization based upon control sources that are fed into the module as well as its stylization state. It is the stylization state that gives the character its individualization. The parameterization of the control module, which encapsulates the information required to produce a motion, is passed on to the animation module. The animation module takes the parameterization as input and using knowledge about how the motion is performed, obtained from the data module, it postures the hierarchical structure of the character over time.

For the system we describe in this article, we will use the motion of a two-legged humanoid gait. However, with the level of abstraction we have imposed on the system, this can easily be replaced with an alternative type of motion, as discussed at the end of this article. The roles of each of the modules within *MovingIK SE* are discussed next.

3.1 Control Module

The parameterization of our system is split into two subgroups. The first specifies the control parameters of the motion, while the second subgroup influences the behavior of the inverse kinematics solution used by the animation module.

The control parameters of the motion are derived from a user-controlled analog joystick whose inputs are used to initially determine the stride length, speed, and direction of travel. The control module subsequently adjusts these basic motion parameters in order to simulate the character's stylization state. This, for example, could be to linearly reduce the maximum speed and stride length in order to simulate a character's fatigue. The way in which the stylization state affects parameterization is discussed later.

The second subgroup of parameters is for weighting the values for stiffening joints so they move less compared to the surrounding limbs. This gives rise to a basic difference in the visual appearance of characters of varying weighted values. The parameters are

determined by the stylization state of the character only. The application of weighted parameters is discussed further in the animation module section of this article.

The stylization state of the control module can be dynamically changed in response to the system's control sources. These can take a variety of forms, including responses to environmental events or an AI engine. For our demonstration in producing stylized motions, we invoke the different states by keyboard input.

As well as the basic control parameters and the weighting values, we have control over hip swing parameters for the walking motion, which we demonstrate. The first parameter controls the amount of rotation in the vertical axis of the hips, which appear as swaying from side to side, with larger values resulting in the character swaying the hips more. The second hip parameter determines the amount of travel there is along the vertical axis, where an increase of this parameter produces a bouncier looking character.

It is the combination of the control and weight parameters that gives rise to realistic individualization of the character.

3.2 Data Module

The data module provides knowledge, in the form of limb degree-of-freedom (DOF) values, on how to perform an action for the animation module. The output format allows us to model the data using a range of techniques including key-framed data, procedural models, and motion-captured data, without affecting the behavior of the other modules in the system. This allows us to choose the optimal data representation for the given scenario, for example, the use of motion-capture data for high detail where storage space is not an issue, compared to procedural models for background characters.

In a previous article we discussed how to adapt a procedural walk model [Meredith and Maddock 2004b]. In the present article we use motion capture data, which gives a more realistic basis. To do this, the data module extracts the DOF values direct from a motion capture file. The motion of the upper part of the body, from the hips upward, remains essentially unaltered, except for a time-warping factor that is used to synchronize the upper and lower body motions.

The motion of the lower body, i.e., from the legs down, is generated based on an adaptation of the original motion data. In our example, we are not just retargetting the walking motion to a new character, but also give our character the ability to change various parameters such as stride length, speed, and direction. To achieve this, we refine the leg motion into a synthesis problem, whereby given a flight path on which the foot should travel, we calculate the unknown hierarchical joint angles using an inverse kinematics solution. However, as we have the original motion data, we make use of this to determine the desired flight path.

Our approach for retrieving the flight path from the motion capture data is based primarily on a gradient analysis technique where, over the course of the original motion, we analyze the height value of one of the character's feet. The first target we look for is a switch from a negative to a positive gradient on a frame that is within 10% of the global minimum height for the foot. This gives us the start of the foot flight where the foot is just about to leave the ground.

From the first target, we progress along the original flight path until we meet a second gradient change from positive to negative, which is on a frame whose height value is within 10% of the global maximum. This second target is the peak of our flight height. Continuing along the original path, we locate the third and final target via one last gradient switch from negative to positive, whose frame height is again within 10% of the global minimum. This technique is demonstrated pictorially in Figure 2.

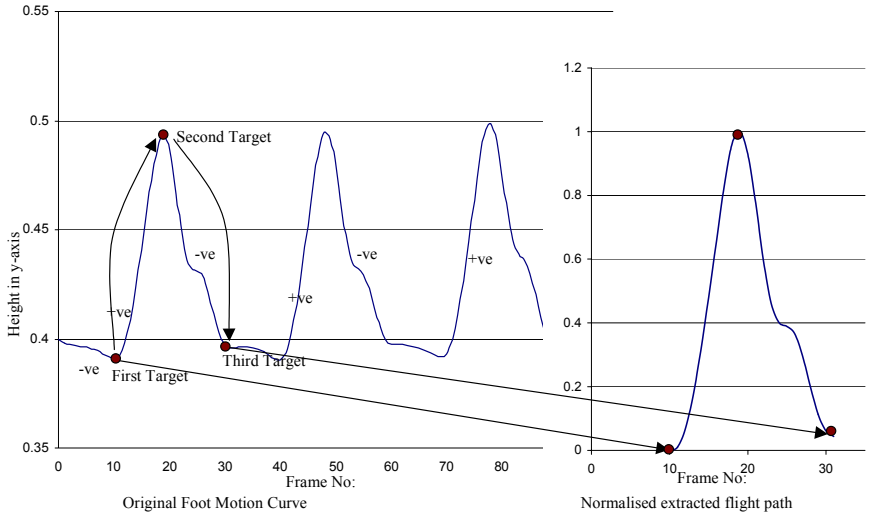


Fig. 2. Gradient-based extraction of foot flight from motion capture data.

We include 10% threshold values when determining the target points in order to reduce the chances of failing to find a complete stride cycle, since using local gradients alone would fail where the motion data temporarily plateaus or the motion data has a little "blip". For example, in Figure 2, near frame 25, the motion data introduces what is almost a stationary point, and had the gradient actually changed at this point, the gradient-analysis technique would have determined this to be the third target, which is obviously not the case.

Having found all three targets, we take the flight curve to be that described between the first and third target points. We subsequently normalize the height values so that they can be mapped onto the new character based on a ratio between the new and original character's leg lengths. We further use the ratio between the characters' leg lengths to initially determine the stride length control parameter.

3.3 Animation Module

Through a combination of information from the control parameters and examination of the surrounding terrain, the animation module determines the extent of the motions' performance. Using knowledge of the motion, the end-effector locations are interpolated over time between the extremities, and thus produce the retargetted motion. The interpolation follows a modified path whose original is obtained from the data module where the end-effectors are positioned along the path through the use of an inverse kinematics solver. This technique allows us to manoeuvre over uneven terrain, including climbing and descending steps. The way in which we adjust the base motion for a walking character is briefly outlined in the following section.

The use of an inverse kinematics solution to configure the character's structure allows us to position end-effector locations precisely. This has the immediate benefit of eliminating visual problems that become apparent when playing pre-scripted animations. However, displaying pre-scripted animations is computationally cheap, so to compete we need to keep the resource demand for the IK solution as low as possible. To this end, we utilize a half-Jacobian-based solver [Meredith and Maddock 2004a], which allows for a more efficient and quicker solution time for IK chains over the traditional, full Jacobian,

Table I. The Two-Stage Walk Cycle: The Initial Configuration Shows the Left Foot in Front and the Right Foot Behind the Body

Stage Description	(a)	(b)
Starting Configuration		
• Left Foot	Both heels and toes are planted on the floor	Toes are planted on the floor
• Right Foot	Toes are planted on the floor	Heel is planted on the floor
Movement	1. Hips move forward, 2. Right heel is advanced forward through the air, 3. Only the left toes remain planted.	1. Hips move forward, 2. Right toes are gravitated towards the floor, 3. Left toes remain planted to the floor

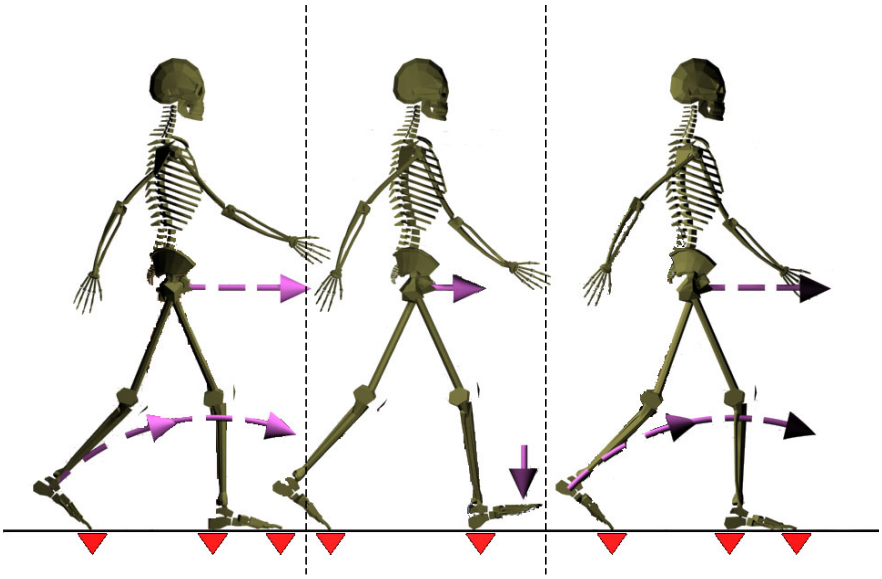
solver. While the full Jacobian IK solver can operate in real-time, the half-Jacobian has a reduced footprint in terms of processor usage, and hence gives a more attractive online animation technique.

The use of a Jacobian-based inverse kinematics solver has the additional benefit of allowing us to take this algorithm and embed a set of dynamic weighting values. These values are the weighting parameters that the control module determines and passes down to this module. The purpose of the weighting values is to give us additional control over the solution that the algorithm produces, which visually generates different inter-limb movements for the same end-effector and root node positions. In effect, we can make use of the inverse kinematics technique to produce a level of character individualization at no additional cost to the core algorithm. This technique is further discussed in the *Weighted Inverse Kinematics* (Section.3.3.2).

3.3.1 *Changing the Base Motion.* There are two cases under which the character can move forward: walking in a straight line or turning. The control module uses input from the control sources to determine the way the character moves. If there is no sideways movement, then the walking forward technique is used; otherwise a turning action is executed.

We split a complete walking cycle into two discrete movements. The first is the actual flight of the foot as the character performs a stride, while the second is a post-flight stage that rolls the foot from a heel-supporting phase to a complete foot-supporting one. The post-flight phase of the walking cycle increases the realistic look of the resulting animation and gives us the ability to model the complete foot as opposed to just the heel. An overview of this procedure is outlined in Table I and Figure 3, where a detailed approach to how we control the character is given in Meredith and Maddock [2004a].

3.3.2 ***Weighted Inverse Kinematics.*** At the center of the animation module is an inverse kinematics engine that postures a character using end-effector locations. We use the half-Jacobian technique [Meredith and Maddock 2004a]. Due to the nature of Jacobian-based inverse kinematic solvers, we are able to predictably modify the degree of change of the different DOFs when configuring a posture, resulting in subtle but individualized results. This, for example, allows us to favor the rate of angle change for the knee over the hip joint.



(a) Start of walk cycle (b) Final part of walk (c) Start cycle again on other foot

Fig. 3. Demonstration of the cycles implemented in our system. Each frame represents the start of the cycle with the arrows pointing in the direction the node will take until it reaches the start of the next part of the cycle. The red triangles represent foot plants of the character.

The Jacobian J at the core of the algorithm is determined from the equation of forward kinematics, given by (1), where θ represents the set of orientation values for a structure, and X is the global position of an end-effector in the hierarchy.

$$X = f(\theta). \tag{1}$$

Taking partial derivatives of (1):

$$dX = J(\theta)d\theta \tag{2}$$

where

$$J_{ij} = \frac{\partial f_j}{\partial x_i}. \tag{3}$$

Rearranging (2):

$$d\theta = J^{-1}dX.$$

Using these equations, we can describe the complete inverse kinematics solver as follows:

1) Calculate the difference between the goal position and the actual position of the end-effector:
   $$dX = X_g - X.$$
2) Calculate the Jacobian matrix using the current joint angles (using (3)).

3) Calculate the pseudo-inverse of the Jacobian:
$$J^{-1} = J^T (JJ^T)^{-1}.$$
4) Determine the error of the pseudo-inverse,
$$error = \left\| (I - JJ^{-1}) dX \right\|.$$
5) If error > *e,* then
$$dX = dX / 2,$$
restart at step 4.
6) Calculate the updated values for the joint orientations and use them as the new current values:
$$\theta = \theta + J^{-1} dX. \quad (4)$$
7) Using forward kinematics, determine whether the new joint orientations position the end-effector close enough to the desired absolute location. If the solution is adequate, then terminate the algorithm otherwise go back to step 1.

For our purposes, we are interested in step 6 of the above algorithm, which is the stage of the algorithm that updates the joint angles within the hierarchical structure. The algorithm, as illustrated above, will distribute the angle changes needed to meet the desired end-effector location evenly over the chain. However, we have rewritten this stage to include a set of dynamic weights that redistribute the contribution each degree of freedom (DOF) has in the resulting motion. We therefore replace (4) with (5) in our implementation, where *W* is a weighting vector.
$$\theta = \theta + WJ^{-1} dX. \quad (5)$$
The weighting vector *W* contains real values between 0 and 1, where smaller values result in less significant changes in the angle compared to larger values in *W* that correspond to bigger angle changes. This principle is illustrated in Figure 4, where the IK solver is applied to a simple hierarchical structure of different weighting values. In Figure.4, we reduced the weighting parameter for the first joint angle, which is at the root, and compared this with an even distribution. As the illustration demonstrates, the joint angle,

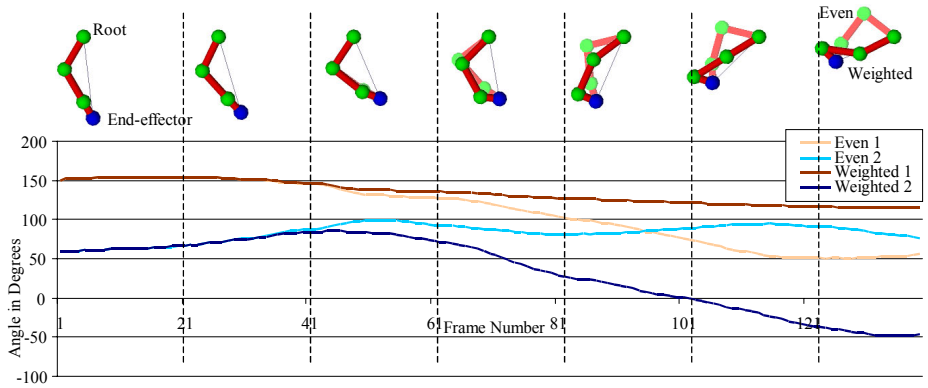


Fig. 4. Application of weighted IK chains on a simple articulated structure. *Top*: Comparison of even (lighter color chain) and weighted (darker color chain) distribution update of angles. The root node angle has a reduced weighting value over the rest of the chain. *Bottom*: Graph of the first two angles in the IK chain working from the root node outwards.

which has a reduced weighting, moves less, therefore other joints in the chain have to move more to meet the desired end-effector position. This is compared to the evenly-weighted IK chain, in which each angle involved in the chain is changed relatively equally.

Although we have only applied a weighting change to one of the angles in Figure 4, the principle of relatively stiffening up joints within an IK chain equally applies when changing multiple weighting values. However, as can be seen from the graph in Figure 4, reducing a weighting on one joint has the effect of indirectly increasing the weights of the remaining joints, since the difference needs to be resolved. This cause and effect result needs to be considered when applying weighting values to an IK chain. Through our experiments, we found that as long as these values are specified relative to each other, the results obtained from the solution exhibit the intended desired properties. If the weights are not determined in a relative manner but instead along an absolute scale, the visual results obtained would not necessarily follow what is expected based on the weighting parameters.

3.3.3 *Changing Weight Parameters to Individualize Characters*. We harness the property of weighed IK chains in the animation module to assist in the production of motions that are individualized to characters based on the stylization state determined by the control module. Although this technique can be applied to a variety of different motions, we will demonstrate how the method lends itself to where limb individualization of masses/muscle tone can be taken into account to determine the weighting values. As an additional effect of individualization, our technique shows good results when applied to simulating injuries, as we demonstrate in the next section**.**

For the subtle changes involved in individualizing a character that is performing a normal motion, we change the weighting parameters only slightly. This has the effect of simulating limb build within the model. For example, large limbs with low muscle tone will have low weights to simulate a sluggish movement, while muscular limbs of the same size will have higher weightings to account for the strength of the muscle.

In order to simulate injuries, as well as adjust the control parameters, we stiffened up the corresponding limb by decreasing the weighting value associated with it. This reflects the fact that the character changes movement in that part of its body where a restriction is introduced by infliction of an injury or to reduce the pain caused by using the limb in a normal motion.

The results of applying different weighting and control parameters to a single base motion are discussed and illustrated in the next section.

4. RESULTS AND DISCUSSION

To illustrate our technique for applying individualizing characteristics to characters in real-time on the basis of motion capture data, we produced a range of different motions by changing only the parameterization that is determined by the control module. The results are obtained by running *MovingIK SE* on a Pentium 4 1.4GHz with a GeForce2 Ultra graphics card. The demonstration animations are achieved in real time running four characters simultaneously in response to the same user input.

Figure 5(a)[1] shows how changing the weighting parameters of a character can be used to produce a slight deviation in the normal walking motion, and hence result in an individualized motion. The motions adapted in Figure 5(a) are portrayed with the

[1] Animations of the stills of Figure 5 are available at http://www.dcs.shef.ac.uk/~mikem/research/ik.html

skeleton bodies, while the original motion capture data is visible as a stick character to the right of the skeletons. Each of the three skeletons in Figure 5(a) is subject to the same control parameters; however, the weighting vector applied to the inverse kinematics solver varies over the skeletons. The red skeleton has an evenly weighted distribution, i.e., all the values are 1, whereas the green and blue skeletons have weighting vectors that stiffen the hip and knee joints, respectively. In the case of the unevenly weighted skeletons, we decreased the respective weightings for the joints by 80% and 20% and left the rest of the weighting values the same as those for the red skeleton.

From the joint angle graph in Figure5(a), the hip angle for the blue skeleton can still be seen to reach a similar maximum and minimum angular measurement as the evenly distributed red character, but it follows a different path over time to achieve this. The effect of using the weighting reduces the rate of angular change for the knee joint for the blue character, and hence, to compensate, the hip angle is changed differently. In comparison, the green skeleton has a stiffened hip joint, and therefore, as can be seen from the joint graph of Figure 5(a), the amount of change for this part of the body is much reduced. In this case, extra movement in the knee joint compensates for the reduced angular change that can be seen in the green skeleton's hip.

As the IK chains get close to being completely extended, as it does at the extents of the walking cycle, the blue weighted version takes on a similar configuration to that of the evenly distributed red one. This is because the possible solutions the IK solver can generate are more tightly packed into a smaller spatial configuration area, so the results look similar in either case. As you would expect, at the start and end of the walk cycles, the postures of the characters appear a close match.

Despite the solution looking similar at the beginning and end of the cycles for the red and blue skeletons (the green one too is very similar but slightly more divergent from the other two), we argue that the differences during the walking phase are enough to demonstrate an individualization of the character, which is achieved by purely applying weighting vectors to the character. The end configurations could be further diversified if we were to adjust the control parameterization and skeletal limb lengths. However, in the results, we have tried to keep as many parameters constant as possible to demonstrate the potential of weighted IK chains. Furthermore, our data and animation modules are additional contributing factors to the similar-looking configurations at the extremities of a cycle because we have specified how a character lands using a two-stage process, as shown in Figure 3, thereby limiting the possible configuration space.

By changing the weighting parameters alone, we are able to generate many individual motions. However, we found that the most natural-looking motions tend to be linked with low variances in the weight vector. Larger variances in the weight vector lead to a noticeable exaggeration in the resulting motion, since, in order to meet an end-effector location, joint angles are not updated significantly until there is no choice. This can be seen by comparing the green and blue skeletons in Figure 5(a), where the green character has an 80% reduction in its weighting value compared to 20% for the blue character. The motions generated with high-variance weighted vectors could account for normal motion in defined cases, but we found it tends to lend itself better to motions that, with a slight adaptation in the other control parameters, simulate the appearance of injuries.

The generation of a walking motion caused by injury is illustrated in Figure 5(b), using both weighted and even distributions over the IK chains. For each of the skeletons in Figure 5(b), the weighting parameters are similar to those in Figure 5(a); however, the weights were only applied to the left leg, which is the one we made limp, and the control parameters were adjusted constantly over the characters. The control parameters were

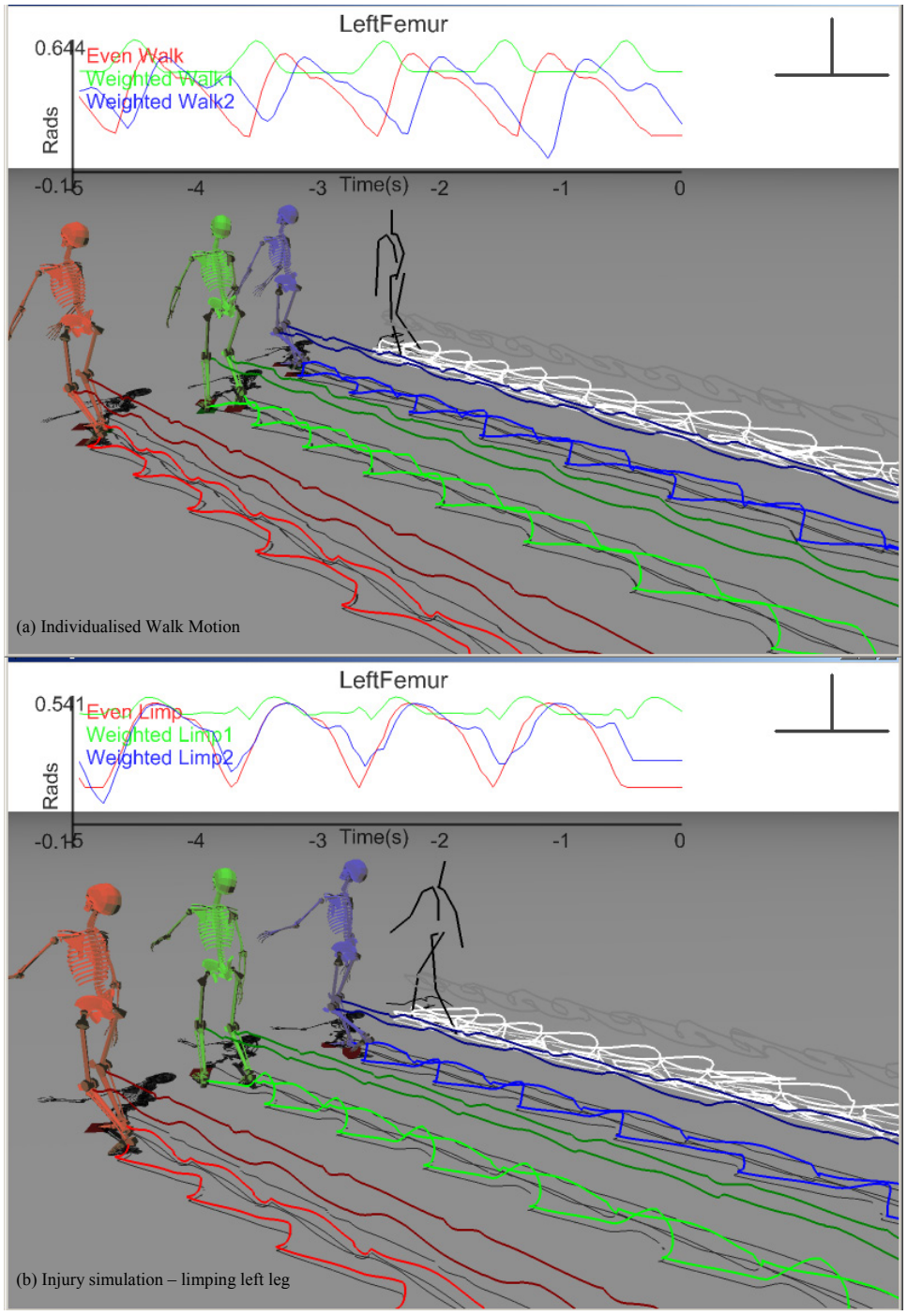


Fig. 5. Application of *MovingIK SE* to adapt original motion capture data to (a) individualize and (b) simulate injury to three different characters with different IK weighting vectors.

adjusted for the left stride in order to decrease the maximum flight height by 50%, to reduce the stride length by 50%, and to increase the speed with which the stride is undertaken by 30%.

Comparing the red skeleton in Figure 5(a) to the one in Figure 5(b), it is clear from the motion trails that simply adjusting the control parameters are enough to visually change the appearance of the walking motion. This is most visually apparent when comparing the motion trails for the limping red character's feet. Here the left-hand side trail barely skims the ground, whereas the right-hand side trail contains much more clearance. Furthermore, the trail for the right foot is much smoother and travels further per stride than the limping left foot; a trait that can be seen in all of the limping skeletons in Figure 5(b). An additional visual effect, not demonstrated in the stills in Figure 5(b), is that the speed at which the stride cycle is undertaken is faster for the injured leg than for the normal walking motion. As illustrated, we can produce a limping motion by simply adjusting the control parameters of the walking motion; but adjusting the weighting parameters can also enhance realism. Similarly to the way in which we applied weightings to normal walking motion, we applied weighting parameters to the green skeleton to stiffen the left hip joint, while the blue skeleton's left knee joint was made stiffer via weighting values. As the joint angle graphs in Figure 5(b) illustrate, applying these weightings noticeably reduces the amount of movement in the limb compared to the evenly distributed limping motion. This is most noticeable in the green skeleton's hip joint because this angle is graphed.

As was the case for normal working motions, the effect of stiffening a joint is to make the remaining angles move more to compensate. This translates into the ability to identify which part of the leg is suffering from the injury. For example, the weighting vector used to simulate the motion of the blue skeleton is tailored to produce an animation that depicts a knee injury. However, when we reset the weight values and decrease the weight value associated with the hip joint, we are able to simulate a hip injury, as with the green skeleton.

The injury position is achieved by maintaining the control parameters, which like the basic process of individualization, demonstrates the usefulness of the weighting values in generating subtle differences between base motions, thereby customising the resulting motion for a specific stylization. This makes it possible to determine where along the leg the injury is being simulated.

As the results demonstrate, the additional factor of weighted IK chains provides a good level of differentiation between the changes in joint angles within the character, which visually introduces subtle changes for motions with the same control parameters. This allows us to spawn many motions, each individualized towards a specific character's attributes, at no extra computational cost to the core IK algorithm.

5. CONCLUSIONS AND FUTURE WORK

The use of real-time inverse kinematics to adapt existing motion-captured animation has the primary advantage of reducing visual artefacts associated with such animations, i.e., the motion is successfully retargeted to the new character and environment. Through the use of weighted IK chains, we have demonstrated an enhancement to the technique that produces richer visual realism at no extra computational cost. Using weighting values, we have shown that it is possible to take a single motion-captured animation and to adjust the motion to different characters; thereby demonstrating a computationally cheap mechanism for producing new retargeted and individualized character animations from a single motion-captured data file. Taking the technique a step further, we have further

shown how the same base representation can be adapted to simulate injury stylization by adding in discrete visual differences.

From the results, we have found that weighting vectors that have small variance values over their elements produce normal but subtly different motions suitable to the individualization of character motions. This level of individualization allows us to control the relative build of the character by assigning comparatively smaller weights to those joints that we expect to change less than others due to muscle structure. This is owing to the direct relationship between the amount of angular change performed during the IK algorithm and the weighting values.

Going beyond basic character individualization, the use of larger variances within the weighting vectors generates exaggerated motions that we used to depict an injured motion. The simulation of injuries is enhanced through the use of changes in the control parameters, and although this has the effect of fundamentally changing the resulting motion, the weighting values give further control in producing a good-looking motion.

We have demonstrated this technique specifically on characters' gaits. However, the idea of adapting the IK result based on weighted chains can apply to any form of posturing using IK. At the heart of this technique, the effect of applying weighting values within the IK chains is to directly affect the rate of change in joint angles. This is a mathematical adjustment on the IK solver itself; therefore anything that makes use of the algorithm can utilize this work. Turning to the field of character animation, weighted IK has most potential where computational costs need to be kept minimal, but enhanced realism is desirable in the resulting motions.

The next application of this technique is to posture the limbs of the upper body where it would be reasonable to assume that the same arguments we presented to demonstrate the application on the legs would also hold for the arms. The application of this algorithm to the upper body will probably produce a much more varied visual result than when applied to the legs, due to the increased number of DOFs available to manipulate via the algorithm. An extended area of application that this algorithm is well suited to is the performance of real-time full-body motion retargetting and individualization.

We acknowledge that this technique will not produce individualization at the level of realism that dynamics-based techniques do; however, we do say that this technique can give a good approximation of the desired results while being computationally cheaper. Although our technique, like any IK solution, is more computationally expensive than applying an unadapted, pre-scripted motion to a character directly, its advantages, namely individualization and stylization, are certainly worth considering and appreciating.

REFERENCES

BROGAN, D. C., METOYER, R. A., AND HODGINS, J. K. 1998. Dynamically simulated characters in virtual environments. *IEEE Computer Graphics and Applications 15*, 5 (1998), 58-69.

DENSLEY, D. J. AND WILLIS, P. J.1997. Emotional posturing: A method towards achieving emotional figure animation. *Computer Animation* (1997).

FEDOR, M. 2003. Application of inverse kinematics for skeleton manipulation in real-time. *Computer Graphics and Interactive Techniques* (2003), 203-212.

GLEICHER, M.1998. Retargetting motion to new characters. In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques*. 33-42.

HODGINS, J. K. AND POLLARD, N. S.1997. Adapting simulated behaviours for new characters. In *Proceedings of the ACM Conference on Computer Graphics*. SIGGRAPH 97. ACM, New York.153-162.

HODGINS, J. K., WOOTEN, W. L., BROGAN, D. C., AND O'BRIEN, J. F. 1995. Animating human athletics. In *Proceedings of the ACM Conference on Computer Graphics*. SIGGRAPH 95. ACM, New York.71-78.

KOMURA, T. AND SHINAGAWA, Y. 2001. Attaching physiological effects to motion-captured data. *Graphics Interface* (2001), 27-36.

KRAUS, M. 2004. Human motion and emotion parameterization. In *Proceedings of the Conference of the Central European Seminar on Computer Graphics.*

MEREDITH, M. AND MADDOCK, S. 2004a. Real-time inverse kinematics: The return of the Jacobian. Tech. Rep. CS-04-06, Dept. of Computer Science, Univ. of Sheffield.

MEREDITH, M. AND MADDOCK, S. 2004b. Individualised character motion using weighted real-time inverse kinematics. *Game-On* (2004).

PARK, J., KANG, Y., KIM, S., AND CHO, H. 1997. Expressive character animation with energy constraints. In *Proceedings of the Compugraphics 97 Conference*. 260-268.

TOLANI, D., GOSWAMI, A., AND BALDER, N. I.. 2000. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphics Models 62,* 5 (2000), 353-388.

UNUMA, M. AND TAKEUCHI, R. 1993. Generation of human motion with emotion. *Computer Animation* (1993), 77-88.

UNUMA, M., ANIYO, K., AND TAKEUCHI, R. 1995. Fourier principles for emotion-based human figure animation. *Computer Graphics and Interactive Techniques* (1995), 91-96.

VICON MOTION SYSTEMS LTD. 2004. http:// www.vicon.com.

WATT, A. AND WATT, M. 1992. *Advanced Animation and Rendering Techniques*. Addison-Wesley, Reading, MA, 1992.

ZHAO, J. AND BADLER, N. I. 1994. Inverse kinematics positioning using nonlinear programming for jiggly articulated figures. *ACM Trans. on Graphics 12*, 4 (1994), 313-336.

Received November 2004; accepted January 2005