INDIVIDUALISED CHARACTER MOTION USING WEIGHTED REAL-TIME INVERSE KINEMATICS

Michael Meredith & Steve Maddock
Department of Computer Science
University of Sheffield
United Kingdom
E-mail: {M.Meredith, S.Maddock}@dcs.shef.ac.uk

KEYWORDS

Character Animation, Character Stylisation, Character Individualisation, Real-time Inverse Kinematics, Weighted IK Chains, Injury Simulation

ABSTRACT

In this paper 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 individualisation 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 individualisation aspect of our technique is very tightly coupled with the inverse kinematics solver, we can achieve both the individualisation and retargetting of characters in real time.

INTRODUCTION

Computers have long been a tool in the construction of character animation starting with the basic key-framing technique used by artists to the full body motion capture used in many applications today. However, regardless of the format the data is held in, it is general practice to simply play back the animation as a pre-scripted entity in real-time applications. This can lead to many unwanted artefacts such as the visually-incorrect appearance of the character sliding across the ground or character-object penetration.

The term retargeting includes the problem of the character sliding across the ground, which is caused by trying to apply a movement captured on one person to another person of a different size. A solution to this particular problem is to store sets of movements for each individual character, which is essentially an off-line retargeting process, albeit a data intensive one. Instead a more flexible online approach to the general retargeting problem is to use inverse kinematics (see (Watt & Watt 92) for a general introduction to IK) to adapt a single set of motions. 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 stylisation 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.

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 individualisation and injury simulation.

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 2004), while the other technique requires the motion to be simulated. The latter can itself be further broken down into several different techniques that including keyframing, inverse kinematics (Tolani *et al* 2000, Zhao & 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 needs 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 and has been successfully tackled in the past with a variety of different techniques include IK (Fedor 2003), spacetime constraints (Gleicher 1998) and dynamics (Hodgins & Pollard 1997). The former techniques tend to be less computational expensive than the latter ones and in particular, the IK algorithm we use in this paper has the ability to perform real-time retargetting in addition to the individualisation we present.

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

Another area of interest for adding stylisation 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 & Takeuchi 1993, Unuma *et al* 1995), energy consumption (Park *et al* 1997) and emotional posturing (Densley & Willis 1997). The latter technique introduces an emotional appearance to the character by constraining joint angles based on the emotional state of the character. Although we do not investigate this in our paper, it would be possible to incorporate this into our work to enhance the individualisation we demonstrate later in this paper, further adding value to our design.

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

CHARACTER INDIVIDUALISATION

Our system, *MovingIK SE*, is comprised of three independent modules that communicate accordingly using a level of parameterisation that allows flexible control over the generated motions. The system's modular design is outlined in Figure 1.1.

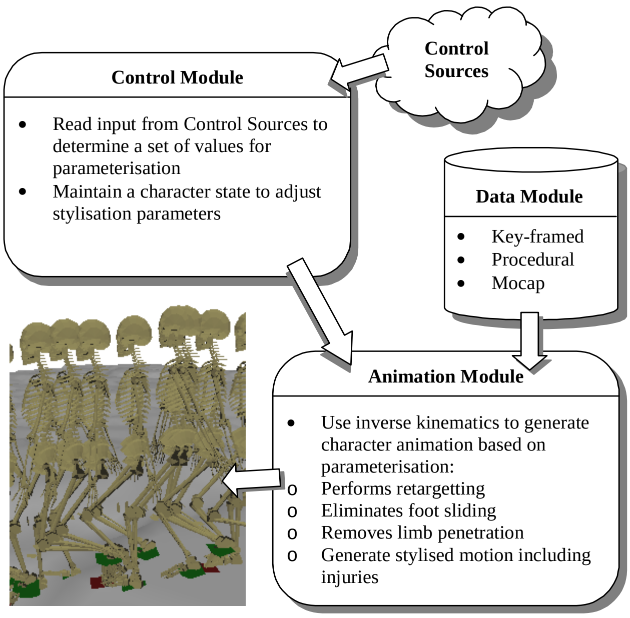


Figure 1.1: Control Structure of *MovingIK SE*.

The Control Module is the top-level component whose purpose is to generate a set of values for the parameterised motion. The Control Module determines the parameterisation based upon Control Sources that are fed into the module as well as its stylisation state. It is the stylisation state that gives the character its individualisation. The parameterisation 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 parameterisation as input and using knowledge about how the motion is performed, which is obtained from the Data Module, it postures the hierarchical structure of the character over time.

For the system we describe in this paper, we will be using 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 the paper. The roles of each of the modules within *MovingIK SE* are discussed in the following subsections.

Control Module

The parameterisation of our system is split into two subgroups. The first subgroup specifies the control parameters of the motion, while the second subgroup influences the behaviour of the inverse kinematics solution used by the Animation Module.

The control parameters of the motion are derived from a user-controlled analogue 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 stylisation state. This, for example, could be to linearly reduce the maximum speed and stride length in order to simulate the fatigue of a character. The way in which the stylisation state affects the parameterisation is discussed later.

The second subgroup of parameters is weighting values that stiffen up joints so they move less compared to surrounding limbs. This gives rise to a basic difference in visual appearance between characters of varying weighted values. These parameters are determined by the stylisation state of the character only. The application of weighted parameters is discussed further under the Animation Module section of this paper.

The stylisation state of the Control Module can be dynamically changed in response to the system's Control Sources. These can take a variety of different forms including responses to environmental events or an AI engine. For our demonstration of producing stylised 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 we demonstrate. The first of these parameters controls the amount of rotation there is about the vertical axis of the hips. This gives the visual appearance of swaying from side to side, with larger values resulting in the character swaying its 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 more bouncy looking character.

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

Data Module

The Data Module provides knowledge, in the form of limb Degree-of-Freedom (DOF) values, about how to perform an action to 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 behaviour 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.

We have based our gait motion data on a mathematical equation which is given in Equation 1.1. A complete cycle ranges from 0 to 3π and the result is the relative height above the ground of the foot. The graphical representation of the procedural stride is illustrated in Figure 1.2.

If $x \leq \pi$

$$Y = 1 - \cos(x)$$

else

$$Y = 1 - \cos\left(\frac{\pi + x}{2}\right) \qquad \textbf{(1.1)}$$

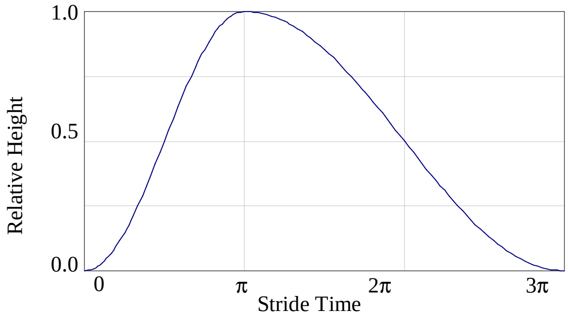


Figure 1.2: Graph of procedural stride of Equation 1.1

Animation Module

Through a combination of information from the control parameters and examination of the surrounding terrain, the Animation Module determines the extents of the motion to perform. Using knowledge about the motion, the end-effector locations are interpolated over time between the extremities and thus produce the retargeted 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 subsection.

The use of an inverse kinematics solution to configure the character's structure allows us to precisely position end-effector locations. This has the immediate benefit of eliminating visual problems that are apparent when playing pre-scripted animations. However pre-scripted animations are computational cheap to display therefore to compete we need to keep the resource demand for the IK solution as low as possible. To this end, we utilise a half-Jacobian-based solver (Meredith & Maddock 2004), which allows for a more efficient and quicker solution time for IK chains over the traditional, full Jacobian 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 nodes positions. In effect, we can make use of the inverse kinematics technique to produce a level of character individualisation at no additional cost to the core algorithm. This technique is further discussed in the *Weighted Inverse Kinematics* subsection.

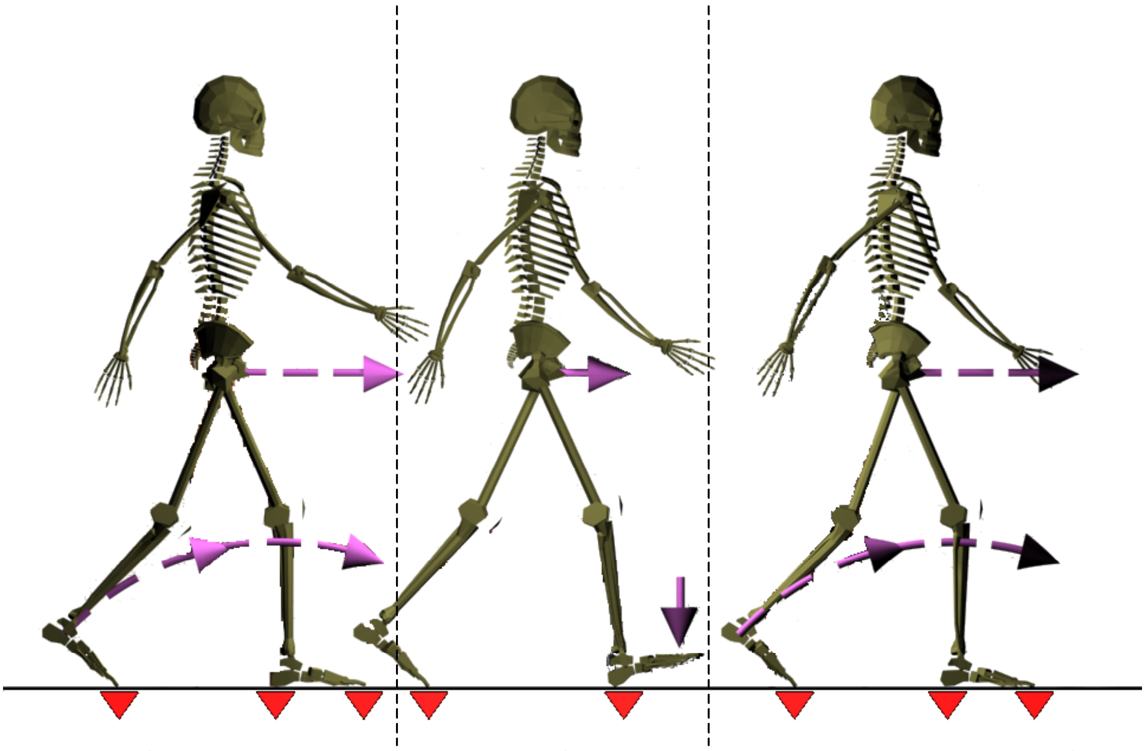
Changing the Base Motion

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

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

Table 1.1: Illustration of the 2-stage walk cycle where the initial configuration is with the left foot in front and the right foot behind the body.



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

Figure 1.3: Demonstration of the cycles implementing in our system. Each frame represents the start of the cycle with the arrows pointing in the direction of travel the node will take until it reaches the start of the next part of the cycle. The red triangles represent plants of the character's limbs.

complete foot supporting phase. The post-flight phase of the walking cycle increases the realistic-looking nature 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 1.1 and Figure 1.3, where a detailed approach of how we control the character is given in (Meredith & Maddock 2004).

Weighted Inverse Kinematics

At the centre of the Animation Module, is an inverse kinematics engine that postures a character using end-effector locations. The algorithm we use is the half-Jacobian-base technique (Meredith & Maddock 2004). Due to the nature of Jacobian-based inverse kinematic solvers, we are able to predictably modify how much different DOFs change when configuring a posture thereby resulting in subtle but individualised results. This, for example, allows us to favour the rate of angle change for the knee over the hip joint.

The Jacobian, J, at the core of the algorithm is determined from the equation of forward kinematics, given by Equation 1.2, where θ represents the set of orientation values for a structure and X is the global position of an end-effector in the hierarchy.

$$X = f(\theta) \quad \textbf{(1.2)}$$

Taking partial derivatives of Equation 1.2:

$$dX = J(\theta)d\theta \quad \textbf{(1.3)}$$

where $$J_{ij} = \frac{\partial f_j}{\partial x_i} \quad \textbf{(1.4)}$$

Rearranging Equation 1.3:

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

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 Equation 1.4)*
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 these as the new current values:*
$$\theta = \theta + J^{-1}dX \quad \textbf{(1.5)}$$
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, it is step 6 of the above algorithm that we are interested in 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

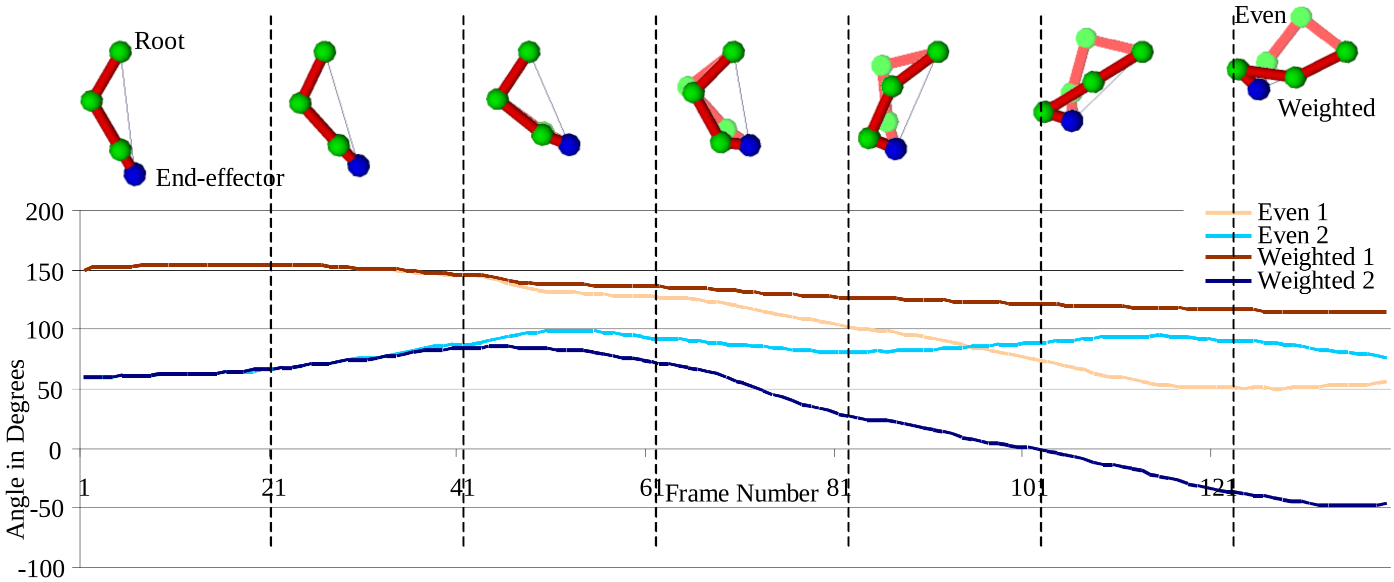


Figure 1.4: Application of weighted IK chains on a simple articulated structure. Top: Comparison of even (lighter colour chain) and weighted (darker colour chain) distribution update of angles – the root node angle has a reduced weighting value over the rest of the chain. Bottom: Graph of the first two angles in the IK chain working from the root node outwards.

resulting motion. We therefore replace Equation 1.5 with Equation 1.6 in our implementation, where *W* is a weighting vector.

$$\theta = \theta + WJ^{-1}dX \qquad \textbf{(1.6)}$$

The weighting vector, *W*, contain real values between 0 and 1 where smaller values result in less significant changes in angle compared to larger values in *W* that correspond to bigger angle changes. This principle is illustrated in Figure 1.4 where the IK solver is applied to a simple hierarchical structure of different weighting values. In Figure 1.4, we have 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 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 of the angles involved in the chain are changed relatively equally.

Although we have only applied a weighting change to one of the angles in Figure 1.4, the principle of relatively stiffening up joints within an IK chain equally applies when changing multiple weighting values. However, as it can be seen from the graph of Figure 1.4, reducing a weighting on one joint has the effector of indirectly increasing the weights of the remaining joints because the difference needs to be resolved. This cause and effect result needs to be considered when applying weighting values to an IK chain. We have found through our experiments that as long as these values are specified relative to each other, the results obtained from the solution exhibit the desired properties intended. If the weights are not determined in a relative manner but instead along an absolute scale, the visual results obtained would not necessarily follow that which is expected based on the weighting parameters.

Changing Weight Parameters to Individualise Characters

We harness the property of weighed IK chains in the Animation Module to assist in the production of motions that are individualised to characters based on the stylisation 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 individualisation where limb masses/muscle tone can be taken into account to determine the weighting values. As an additional effect of individualisation, our technique shows good results when applied to simulating injuries as we demonstrate in the next section.

For the subtle changes involved in individualising a character 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 would have low weights to simulate a sluggish movement while muscular limbs of the same size would have higher weightings to account for the strength of the muscle.

In order to simulate injuries, as well as adjusting the control parameters, we stiffen up the corresponding limb by decreasing the weighting value associated with it. This reflects the fact that the character changes the movement in the part of their body where a restriction is introduced by the infliction or in order to reduce the resulting pain that would result from using the limb in normal motion.

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

RESULTS & DISCUSSION

To illustrate our technique of applying individualisation characteristics in real-time to characters, we produced a range of different motions by changing only the parameterisation 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 4 characters simultaneously in response to the same user input.

Figure 1.5(a, b)[1] shows how changing the weighting parameters of a character to produce a slight deviation in the normal walking motion generates an individualised result. To produce the weighted walking motion of Figure 1.5b we decreased the weighting value at the hip joints by 20% and left the rest of the weights the same as used in Figure 1.5a. From the graph in figure 1.5e, the weighted chain knee joint still reaches a maximum and minimum angular measurement as in the evenly distributed walk; however it follows a different path over time to achieve this. The effect of using the weighting reduces the speed with which the hip angle changes therefore the knee angle is changed more to compensate. This can be seen in the middle frames of Figure 1.5a and 1.5b where the character's left knee is further forward in the evenly distributed solution compared to the weighted version for the same point in time.

As the IK chain gets close to being completely extended, as it does at the extents of the walking cycle, the weighted version takes on a similar configuration to that of the evenly distributed 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. Figure 1.5a and 1.5b illustrate this where it can be seen in the first and last set of frames that the configurations are virtually identical.

Despite the solution looking similar at the beginning and end of the cycle, we argue that the differences during the walking phase are enough to demonstrate an individualisation 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 parameterisation 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 1.3, thereby limiting the possible configuration space.

By changing the weighting parameters alone we are able to generate many individual motions. However, we have found that the most natural-looking motions tend to be linked with low variances in the weight vector used. Larger variances in the weight vector lead to noticeable exaggeration in the resulting motion because joint angles are not updated significantly until there is no choice in order to meet and end-effector location. The motions generated with high-variance weighted vectors could account for normal motion in defined cases however 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 an injured walking motion is illustrated in Figure 15(c, d), with both the use of weighted and even distribution over the IK chains. In both of these illustrations, the control parameters have been similarly adjusted from the normal walking motion. The control parameters are adjusted for the left stride in order to decrease the maximum flight height by 50%, reduce the stride length by 50% and increase the speed with which the stride is undertaken by 30%.

Comparing Figure 1.5a and Figure 1.5c, it is clear that simply adjusting the control parameters is enough to visually change the appearance of the walking motion. This is most visually apparent between the middle frames of the two motions in Figure 1.5. Between these frames, the effect of reducing the flight height of the foot is clear to see, and in the latter frames of the motion it is also visible that the left foot does not travel as far forward as in the normal walking motion. An additional visual effect that is not demonstrated in the stills of Figure 1.5 is that the speed with which the stride cycle is undertaken is faster for the injured leg than for the normal walking motion.

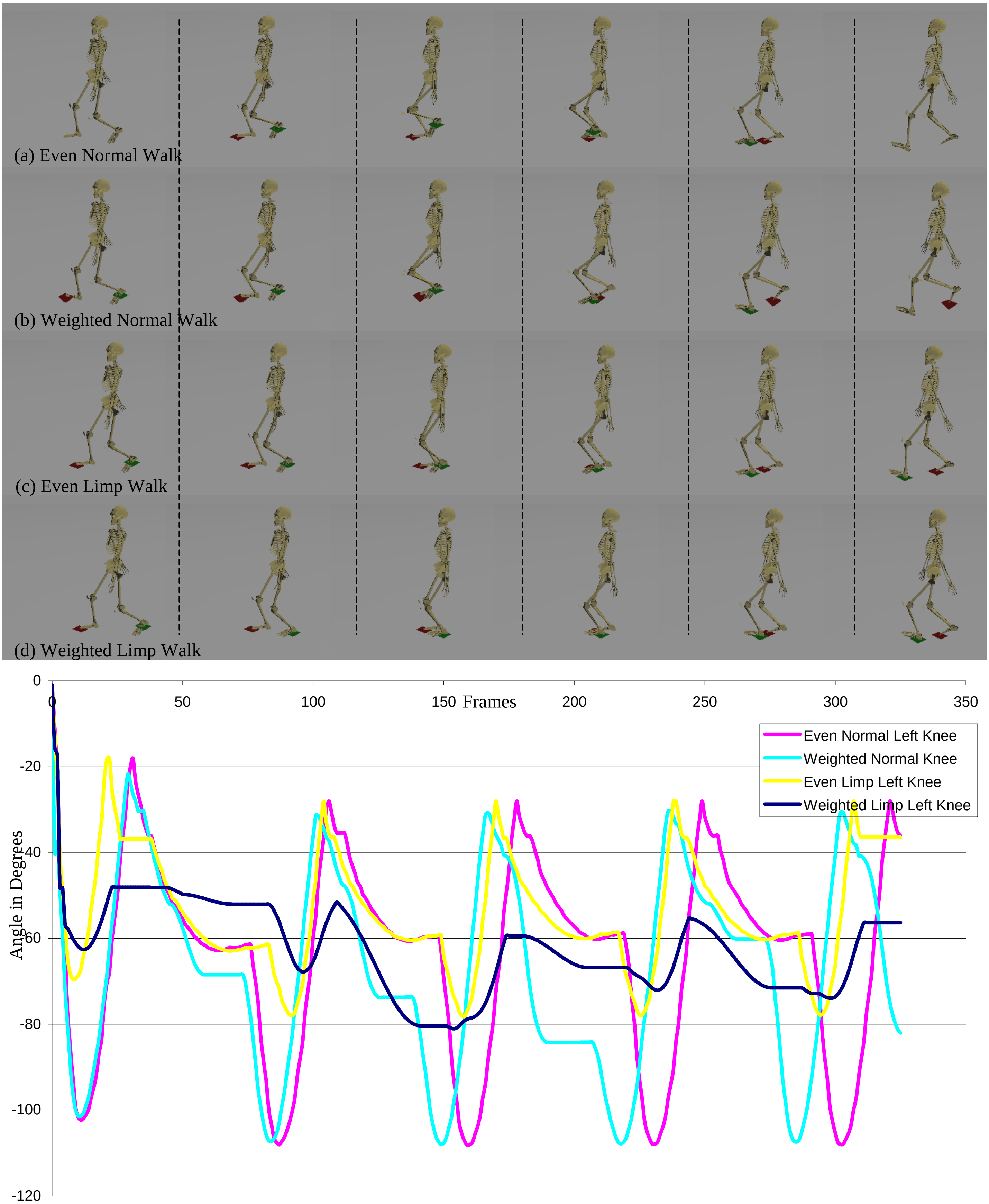
As illustrated, by only adjusting the control parameters of the walking motion we can produce a limping motion. However the realism can be enhanced by additionally adjusting the weighting parameters. This is illustrated in Figure 1.5d. The weight parameters we use to enhance the limp in Figure 1.5d impose a stiffening effect on the left knee, which is achieved by reducing the weighting value for this joint by 90%. As Figure 1.5e illustrates, the effect of applying this weight is to noticeably reduce the amount of movement in the limb compared to the evenly distributed limp motion. This visually translates well into the resulting motion which can be seen when comparing the frames of Figure 1.5c and Figure 1.5d. As these frames show, in order to compensate for the reduced movement in the knee, the hip joint of the weighted motion changes comparatively more than for the evenly distributed one. This is illustrated when comparing the absolute position of the knee joint between the corresponding frames.

The weighting vector used to simulate the motion of Figure 1.5d 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. This is achieved by maintaining the control parameters, which like the basic process of individualisation, demonstrates the usefulness of the weighting values in generating subtle differences between base motions thereby customising the resulting motion for a specific stylisation. This makes is possible to determine where the injury is being simulated along the leg.

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

---

[1] Full animations of the stills of Figure 1.5 are available at http://www.dcs.shef.ac.uk/~mikem/research/ik.html



(e) Graph of left knee joint angle for the walking characters (a)-(d) over 5 strides

Figure 1.5: Demonstration of using weighted chains for individualisation (b) and injury (d) simulation compared to the even distribution of joint changes for the same motions (a) & (c) respectively. Images of (a)-(d) are over a single stride of the left foot.

CONCLUSIONS & FUTURE WORK

The use of real-time inverse kinematics to animate a character has the primary advantage of reducing visual artefacts associated with pre-scripted animations. 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 representation of a motion and to adjust the motion to different characters, thereby demonstrating a computation ally-cheap mechanism for producing individualised character animations. Taking the technique a step further, we have further shown how the same base representation can be adapted to simulate injury stylisation 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 individualisation of character motions. This level of individualisation allows us to control the relative build of the character by assigning comparatively smaller weights to those joints we would expect to change less than others due to muscle structure. This is due to the direct relationship between the amount of angular change performed during the IK algorithm and the weighting values.

Going beyond the basic character individualisation, the use of larger variances within the weighting vectors generates exaggerated motions that we have 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 character 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 utilise 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 upper body limbs where it would be reasonable to assume that the same arguments we have presented to demonstrate the application on the legs would also hold for the arms. The application of this algorithm to the upper body would probably produce a much more varied visual result than with the legs because of the increased number of DOFs available to manipulate with the algorithm. An extended area of application that this algorithm would be well suited to is that of performing real-time full body motion retargeting and individualisation.

We acknowledge that this technique will not produce individualisation to a level of realism that dynamics-based techniques can, however we propose that this technique can give a good approximation to the desired results whilst being computational cheaper. Whilst our technique, like any IK solution, is more computationally expensive than directly applying an unadapted, pre-scripted motion to a character, its advantages, namely individualisation and stylisation, are certainly worth appreciating.

REFERENCES

Brogan, D. C., Metoyer, R. A., and Hodgins, J. K., “Dynamically Simulated Characters in Virtual Environments”, IEEE Computer Graphics and Applications, Vol. 15, No. 5, p. 58-69, 1998

Densley, D.J., Willis, P.J., “Emotional Posturing: A Method Towards Achieving Emotional Figure Animation”, Computer Animation 1997.

Fedor, M., “Application of Inverse Kinematics for Skeleton Manipulation in Real-Time”, Computer Graphics and Interactive Techniques, p. 203-212, 2003

Gleicher, M., “Retargetting Motion to New Characters”, International Conference on Computer Graphics and Interactive Techniques, p. 33-42, 1998

Hodgins, J. K., Pollard, N. S., “Adapting Simulated Behaviours for New Characters”, ACM Siggraph 97, Computer Graphics Proceedings, p. 153-162, 1997

Hodgins, J. K., Wooten, W. L., Brogan, D. C., O’Brien, J. F., “Animating Human Athletics”, ACM Siggraph 95, Computer Graphics, p. 71-78, 1995

Komura, T., Shinagawa, Y., “Attaching Physiological Effects to Motion-Captured Data”, Graphics Interface 2001, p. 27-36, 2001

Kraus, M., “Human Motion and Emotion Parameterization”, Central European Seminar on Computer Graphics, 2004

Meredith, M., Maddock, S., “Real-Time Inverse Kinematics: The Return of the Jacobian”, Technical Report No. CS-04-06, Department of Computer Science, The University of Sheffield, 2004

Park, J., Kang, Y. Kim, S., Cho, H., “Expressive Character Animation with Energy Constraints”, Compugraphics 97, p. 260-268, 1997

Tolani, D., Goswami, A., Balder, N.I., “Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs”, Graphics Models, Vol. 62, No. 5, p.353-388, 2000

Unuma, M., Takeuchi, R., “Generation of Human Motion with Emotion”, Computer Animation 93, p. 77-88, 1993

Unuma, M., Aniyo, K., Takeuchi, R., “Fourier Principles for Emotion-Based Human Figure Animation”, Computer Graphics and Interactive Techniques, p. 91-96, 1995.

Vicon Motion Systems Ltd, http:// www.vicon.com, 2004

Watt, A., Watt, M., “Advanced animation and rendering techniques”, Addison-Wesley, 1992

Zhao, J., Badler, N. I., “Inverse Kinematics Positioning Using Nonlinear Programming for Highly Articulated Figures”, ACM Transactions on Graphics, Vol. 12, No. 4, p. 313-336, 1994