DOI: 10.1111/j.1467-8659.2008.01177.x

COMPUTER GRAPHICS *forum*
Volume 27 (2008), number 6 pp. 1677–1686

Exaggerating Character Motions Using Sub-Joint Hierarchy

Ji-yong Kwon and In-Kwon Lee

Yonsei University, Seoul, Korea
mage@cs.yonsei.ac.kr; iklee@yonsei.ac.kr

Abstract

Motion capture cannot generate cartoon-style animation directly. We emulate the rubber-like exaggerations common in traditional character animation as a means of converting motion capture data into cartoon-like movement. We achieve this using trajectory-based motion exaggeration while allowing the violation of link-length constraints. We extend this technique to obtain smooth, rubber-like motion by dividing the original links into shorter sub-links and computing the positions of joints using Bézier curve interpolation and a mass-spring simulation. This method is fast enough to be used in real time.

Keywords: cartoon stylization, motion exaggeration, sub-joint hierarchy

ACM CCS: I.3.7 Computer Graphics: *Three Dimensional Graphics and Realism*

1. Introduction

A lot of research has been done on creating realistic motion data for character animation. The motion graph [KGP02] and physics-based character animation [LHP05] are two examples of methods that have been developed to meet this objective, and they have produced great results. While such methods are excellent in generating realistic animations, they are not suitable for producing cartoon-style animation which has a different nuance. Traditional keyframe techniques are, of course, frequently used to animate cartoon characters, but these methods involve difficult and tedious work, even when used by a skillful animator. A way of converting motion capture data to cartoon-style character motion would therefore be very useful.

The principles of traditional cartoon animation have been widely introduced into computer graphics [Las87, Wil01], and some mathematical techniques for non-photorealistic animation have been developed, such as squash and stretch [CPIS02, HHD04], and anticipation and follow-through [KCSL06, WDAC06].

We approach the problem differently, and generate cartoon character motions by exaggeration. The key idea of our method is to exaggerate a captured motion even though the resulting motion violates link-length constraints, because we see that exaggerated motion is the basis of a lot of cartoon animations.

Figure 1 shows some examples of the traditional cartoon animations. We see that the chest and the left arm of the human character are temporarily stretched, and neither of the characters satisfies their link-length constraints, nevertheless the exaggerated motions represent the situation nicely. Our method achieves a similar effect through the creation of a sub-joint hierarchy which is a superset of the original joint hierarchy of the character, and then by smoothly exaggerating the motion using trajectory-based motion exaggeration and physical simulation.

Our work includes the following two contributions:

- We introduce a trajectory-based technique to motion exaggeration. Exaggerating joint angles can cause unexpected results due to singularities [LS01], but our method does not suffer from the singularity problem because it exaggerates the spatial joint trajectory. Moreover, our method is as simple as other joint-angle exaggeration techniques [BW95, KCSL06, LS01, WDAC06].
- We suggest a new approach that modifies the character's joint hierarchy. Because most methods for editing motions modify only joint configurations and try to satisfy

Submitted March 2008
Revised May 2008
Accepted June 2008

Figure 1: *Examples of the traditional cartoon animations: in this sequence of frames, note how the chest and the left arm of the character are temporarily stretched.*

the joint hierarchy as a hard constraint, they cannot exaggerate the motion beyond the link-length constraints while still achieving smooth poses. We modify both the joint configurations and the joint hierarchy, so that we can achieve a final motion that has stretched links and rubber-like poses. Results show that our method works well even though it often violates joint hierarchy constraints such as the lengths of links and the number of joints.

The rest of this paper is organized as follows. We review related work in Section 2, and present our algorithm briefly in Section 3. Section 4 describes our trajectory-based technique for motion exaggeration and the rubber-joint group (RJG) scheme that makes the exaggerated motion smooth is presented in Section 5. In Section 6, several simple ways of post-processing the results from our method are introduced. We discuss our results in Section 7, and then draw some conclusions.

2. Related Work

We can segment the extensive work that has been done on processing animation data into photorealistic and non-photorealistic animation approaches.

In the field of photorealistic animation approaches, stylizing character motions [ABC96, GMHP04, HPP05, LGXS03, MK05, UAT95] is related to our work, because a cartoon animation is obviously a form of stylized motion. Stylization has achieved some good results, but not specifically for cartoons. Retargetting the motion to another character [CK00, Gle98, SLSG01, TK00] considers the problem that maps motion of one character into another character with different link lengths, which is similar to our exaggeration problem. However, we also consider various link lengths with respect to time for exaggeration. Harrison *et al*. [HRvdP04] treated the perceptual problem for changing link lengths of a character. Mohr and Gleicher [MG03] proposed the skinning method that automatically adds joints into the original character to interpolate rotations without collapsing. Similarily, our approach uses additional joints to mimic rubber-like bending behaviours.

Work on non-photorealistic animation can be classified into view-dependent approaches, reusing cartoon animation data, and techniques that follow the principles of traditional animation. Rademacher [Rad99] observed that objects in cartoons can have different representations that depend on the viewing direction, and he proposed a view-dependent technique that uses linear interpolation. He also showed that his approach could be applied to animation. A similar approach has been applied to character animation [CKB04]. Bregler *et al*. [BLCD02] attempted to capture cartoon animations and reuse them. They also tried to capture a squash-and-stretch effect from the cartoon video using affine transformations; however, this method does not generate the bending effect of rubber bodies. De Juan and Bodenheimer [dB06] interpolated keyframes captured from cartoons to make new cartoon animation sequences.

Many techniques that follow traditional animation principles simulate squash and stretch [CPIS02, HHD04], which is the most significant shape distortion in cartoon animation [Las87]. Kim *et al*. [KCSL06] automatically generated anticipation and follow through motions in character animation.

Signal processing techniques are significantly related to exaggerated motions. Bruderlin and Williams [BW95] applied multiresolution analysis to each joint channel of a character motion and derived several new motion editing methods. Lee and Shin [LS01] proposed the rotation-invariant multiresolution analysis using quaternions, which do not suffer from the singularity problem of joint angles. More recently, Wang *et al*. [WDAC06] have applied the Laplacian of a Gaussian filter to general animation signals to generate anticipation and follow-through effects. Such methods are simple and fast, and produce exciting exaggerated motions.

Many of the methods briefly reviewed above get good results in either photorealistic or non-photorealistic animation, but have the limitation that they cannot violate a character's joint hierarchy. In general, a joint hierarchy maintains a link at its pre-defined length, which limits the extent of any exaggeration. However, in traditional cartoons we can easily find examples where link-length constraints are violated.

3. Algorithm Overview

Figure 2 provides an overview of our algorithm, which consists of two steps. First, trajectory-based motion

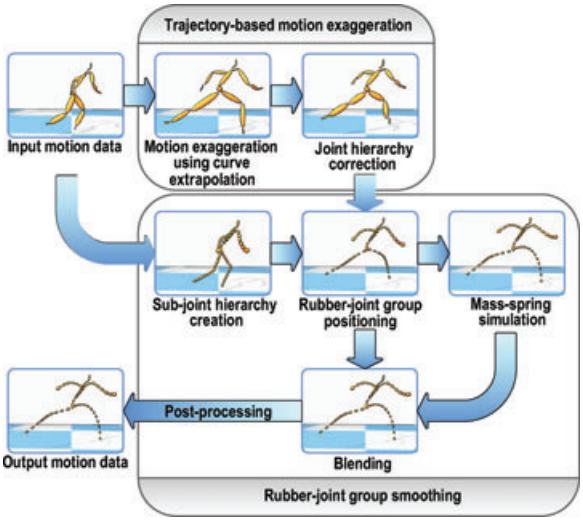

Figure 2: *Overview of our algorithm.*

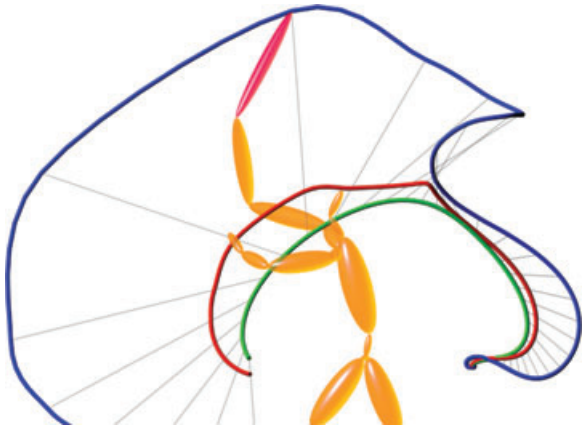

Figure 3: *Motion exaggeration using curve extrapolation: the red curve is the original trajectory $j_k(t)$; the green curve is the average curve $a_k(t)$; and the blue curve is the extrapolated curve $e_k(t)$.*

exaggeration is applied to make a distorted motion which involves the stretching of links. Next we divide the links of the original joint hierarchy into small unit joints, which we call sub-joints, and use them to mimic bending effect of rubber. This sub-joint hierarchy can be precomputed, and we can also get a more interesting motion by adding in a mass-spring simulation. We use basic skeleton representation for a result motion, because it is the most frequently used method in character animation, and can be easily integrated to other systems.

4. Trajectory-Based Motion Exaggeration

4.1. Motion exaggeration using curve extrapolation

Our method exaggerates not joint angles, but joint trajectories. The editing of Euler angles has often been used as a motion editing technique [BW95, WDAC06], but an Euler angle parameterization has to be handled separately. Quaternion-based motion editing [KCSL06, LS01] does not have a singularity problem, but we cannot get a result motion with stretched links using this method because exaggerating the joint configurations does not affect the link-length constraints. Instead, we exaggerate the trajectory of each joint in three-dimensional space, which allows us to stretch some links.

Let $j_k(t)$ denote a trajectory curve of the k^{th} joint J_k with time parameter $t(\in [0, T])$. An exaggerated version $e_k(t)$ of $j_k(t)$ can then be computed as follows:

$$e_k(t) = \omega j_k(t) + (1 - \omega)a_k(t), \tag{1}$$

$$a_k(t) = j_k(t) * g_\sigma(t), \tag{2}$$

$$g_\sigma(t) = e^{\frac{t^2}{(2\sigma^2)}}/(\sigma\sqrt{2\pi}). \tag{3}$$

$g_\sigma(t)$ is a Gaussian function with standard deviation σ. Thus, $a_k(t)$, the convolution of $j_k(t)$ and $g_\sigma(t)$, becomes the local spatial average curve of the joint trajectory $j_k(t)$. $e_k(t)$ is an extrapolation of $j_k(t)$ away from $a_k(t)$, where ω (> 1) is a user-defined parameter. Note that both σ and ω are control parameters determining the magnitude of the exaggeration. We will use $\omega = 5$ for most of experiments in this paper.

A point $a_k(t_0)$ represents the average position of the joint at a specific time t_0. If the actual position is much different from the average at time t_0, then the distance between $j_k(t_0)$ and $a_k(t_0)$ will be large and so the extrapolated point $e_k(t_0)$ will be located far from the point $j_k(t_0)$. Consequently, the extrapolated curve $e_k(t)$ exaggerates the motion by a distance equal to the difference between the actual and the average motion.

By way of analogy, facial caricatures exaggerate the shape of the part of a face which is already different from the average [Red84]. This procedure has been applied to motion exaggeration in the field of visual perception [HPK*02]. Unuma *et al.* [UAT95] used the interpolation and the extrapolation of a normal and a strong motion in order to generate rich variations of human motions. In a similar way, we use a local average curve as the normal situation, which is then compared with the joint trajectory and our extrapolation scheme can then accentuate the difference between the actual and normal trajectories. Figure 3 shows an example of a throwing motion. We can see how deviations from the normal are emphasized by our technique.

Of course the exaggerated motion does not satisfy the original link-length constraints, which is what we expected

```
while error > threshold
  error ← 0
  for t ← 0 to T
    frameError ← 0
    for each l(J_p,J_c) ∈ L
      length ← |v(J_p,J_c)|
      w ← e_p(t) − e_c(t)
      m(J_p,J_c) ← δ · length · w/|w|
      frameError ← frameError + (1 − δ) ∗ length
    end for
    for each l(J_p,J_c) ∈ L
      e_c(t) ← e_c(t) + m(J_p,J_c)
    end for
    error ← error + frameError/T
  end for
end while
```

Figure 4: *Joint hierarchy correction algorithm.*

to happen. However, one may want a motion that does not violate the original link-length constraints too much. In that case, we can correct the exaggerated motion using the algorithm that will be introduced in the next section.

4.2. Fast joint hierarchy correction

To keep the motion from stretching the original link lengths too far, we may need to correct the motion. Let J_k be an arbitrary joint and let $\mathcal{J} = \{J_k\}$ be the set of all the character's joints. Let $l(J_p, J_c)$ be the relation between a parent joint J_p and its child joint J_c, so that $L = \{l(J_p, J_c)\}$ is the set of all such relations, and $\mathbf{v}(J_p, J_c)$ is a link vector from J_p to J_c. For all the relations in L, we want to modify the positions of the joints so that the distance between J_p and J_c comes closer to the length of $\mathbf{v}(J_p, J_c)$.

Figure 4 depicts our algorithm to solve this problem. T is the number of frames in the motion, and δ is a small coefficient that preserves the distortion of the motion during each iteration. The algorithm first measures how much the motion exceeds its link-length constraints, then corrects each position of the joint with the vector $\mathbf{m}(J_p, J_c)$, from the origin of the current joint to the origin of its parent, iteratively. Each joint moves towards its parent, eventually root joint, and the total length of all links almost decreases after each iteration, thus the algorithm halts stably. In addition, we can easily control the amount of exaggeration by modifying the error threshold of the algorithm, because this value gives an intuitive way to control the total length of links. We could use inverse kinematics [TK00] to solve this problem but, as we do not have to satisfy the link-length constraints exactly, our algorithm is perfectly adequate, and runs faster. Figure 5 shows how the exaggeration of a throwing motion is controlled by each successive iteraion of our algorithm.

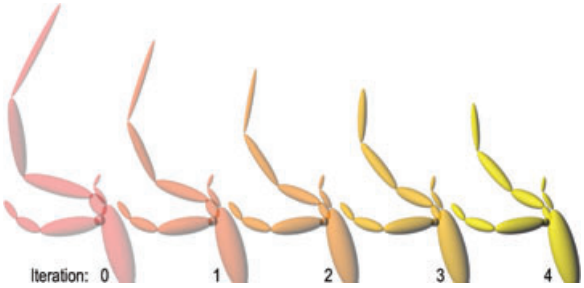

Figure 5: *Fast joint hierarchy correction: an excessively violated link length (in red) is iteratively corrected to an acceptable link length (in yellow).*

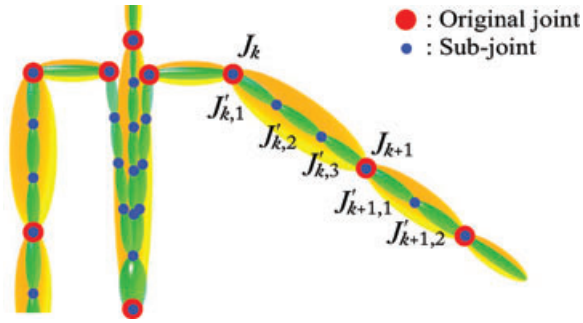

Figure 6: *Creation of a sub-joint hierarchy.*

5. Rubber-Joint Group Smoothing

5.1. Sub-joint hierarchy creation

To apply the bending behaviour of a rubber rod to the links of the character, we define a new sub-joint hierarchy. We subdivide the link at each joint into small sub-links of equal length, and then create a new sub-joint at the end of each sub-link. The positions and orientations of sub-joints can easily be derived from the original joints. Let $\mathcal{J} = \{J_k\}$ be the set of original joints. Let a joint J_k be subdivided into m_k sub-joints, as shown in Figure 6, and let the sub-joints be written $J'_{k,1}, J'_{k,2}, \ldots, J'_{k,m_k}$. The set $\mathcal{J}'_k = \{J'_{k,l}, 1 \le l \le m_k\}$ consists of all sub-joints of J_k, and the new joint set $\mathcal{J}' = \bigcup \mathcal{J}'_k$ is the union of all sets of sub-joints.

5.2. Rubber-joint group positioning

We define a RJG as follows:

- An end-effector is a member of an RJG.
- If a parent joint has only one child joint and that child joint is a member of an RJG, then the parent joint is also a member of that RJG.

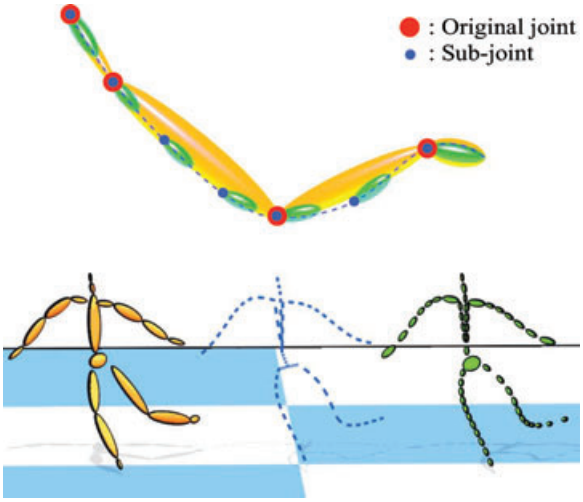

Figure 7: *Rubber-joint group positioning: the positions of the original joints (red circles) are interpolated using Bézier curve interpolation (blue dashed curve), and the positions of sub-joints (blue circles) are evaluated using the resulting curve.*

If the character does not have fingers or toes, we can group the serial links of each limb into one RJG. We will use the RJGs to compute the position of each sub-joint.

To convert a simply exaggerated motion into a rubber-like motion, we interpolate the positions of all the original joints in each RJG for each frame by Bézier curve interpolation [CRE01], for which we use a chord-length parameterization based on the original link length. The parameters of the sub-joints and the length of the links between them then can be easily computed by positioning each sub-joint at a point on the interpolated curve with the appropriate parameter. Figure 7 shows some examples of output from our positioning algorithm. If any sub-joints are not on any RJGs, we use their original positions.

5.3. Mass-spring simulation

We use a mass-spring simulation [WB97] to generate follow-through from an exaggerated motion. Each sub-joint and sub-link can be utilized for constructing a mass particle and spring in a mass-spring system. Because the mass-spring system should be controlled to generate appropriate follow-through motions for a given motion, we use a suspended mass-spring system described below.

Figure 8 describes a structure of our mass-spring system. First, we set an infinite-mass particle on each sub-joint, which is called 'the sub-joint particle'[1] (shown as an orange sphere in the figure.) Another unit-mass particle which is named 'the suspended particle' is set on each sub-joint (shown as a cyan sphere in the figure.) Then, a sub-joint particle and a suspended particle which are located at same position are linked by a zero-length spring, and two suspended particles whose associated sub-joints are linked in a sub-joint hierarchy are also linked by a spring that has the same length as its associated sub-link. Our suspended mass-spring system was inspired by the work of von Funck and his colleagues [vFTS07]. However, there are some differences between our model and von Funck's model. von Funck's model was designed for generating secondary actions of general mesh models. Thus, users should manually set the position of each mass particle, so each particle moves separately. However, our model is for generating follow-through motions of articulated character animation, so it can be automatically set from a given joint hierarchy and moves systemically.

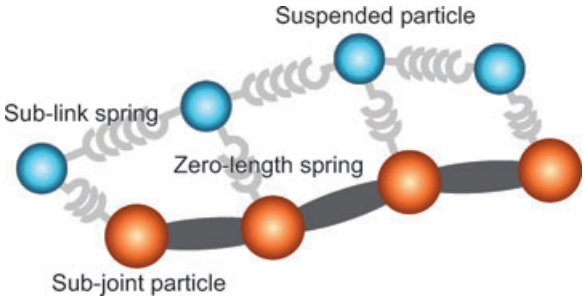

Figure 8: *A structure of a suspended mass-spring system.*

In the simulation process, positions and velocities of all mass particles in the system are first initialized as those of its associated sub-joint of an exaggerated motion. Then, suspended particles are simulated by considering spring forces and drag forces, while states of sub-joint particles are updated in accordance with the exaggerated motion. Finally, we can get the exaggerated motion that is added to appropriate follow-through motions by linearly blending two positions of each sub-joint particle and suspended particle. Using our method, the follow-through motions are effectively generated because the suspended particles always follow sub-joint particles and generate physical effects from spring forces. We can control the stiffness of follow-through motions by adjusting the hook coefficients of springs, and can also control the amount of follow-through motions by determining the blending coefficient between positions of sub-joint particles and suspended particles. Figure 9 shows some snapshots captured from our experimental results for a suspended mass-spring system. We can observe that the simulated motion looks more dynamic and interesting than the simply exaggerated motion.

[1] Note that the 'infinite-mass' particle is the mathematical trick to make its acceleration to zero from the equation $a = f/m$.

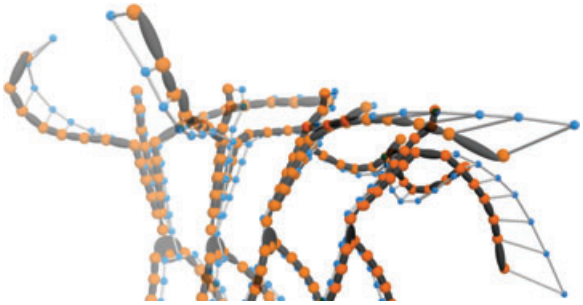

Figure 9: *An example of a suspended mass-spring simulation for throwing motion.*

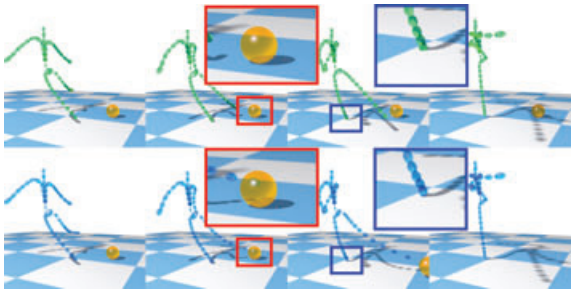

Figure 10: *An example of our experimental result by using the curve deformation method. Top row: an exaggerated kicking motion. Bottom row: a post-processed motion. Notice that the post-processed motion is kicking the yellow ball exactly (marked as red boxes) and keeping the right foot on the floor (marked as blue boxes.)*

6. Post-Processing

The motion produced by our exaggeration and group-smoothing techniques consists of position data at each sub-joint. This must be converted to a general skeleton motion with orientation data. We compute the orientation of each sub-joint by applying two rules: first, we compute the orientations of sub-joints that are located at the positions of the original joints, making these orientation as similar to the orientations of the original joints as possible; then, we determine the orientations of the other sub-joints using Slerp interpolation [Sho85] between the sub-joints that we have already computed. As the new motion has variable link lengths, all six degrees of freedom of each sub-joint must be determined.

As the trajectories of end-effectors are changed when the motion is exaggerated, some post-processes are needed to satisfy kinematic constraints such as foot contact constraints. Such constraints can be described as space-time constraints which mean that a trajectory of an arbitrary end-effector should pass through a specific position at a specific time. We can solve these problems by using the inverse kinematics techniques. However, we can also use the curve editing techniques by considering serial chains of sub-joints as a curve because we do not have to keep the link-length constraints for a character. We experimentally exploited the discrete curve deformation method proposed by Nealen *et al*. [NISA07] to modify trajectories of end-effectors and their corresponding serial chains of sub-joints. Figure 10 shows an example of our experimental result. The exaggerated kicking motion is modified to keep its feet on the floor and to kick the ball at the specific position. We can observe that the motion satisfies the constraints described above by using the curve deformation method.

We cannot use the skinning information that was originally used to generate the characters from its skeleton because a new joint hierarchy has been generated by the exaggeration technique. We obtain new skinning data by modifying the original skinning data to suit the changed joint hierarchy.

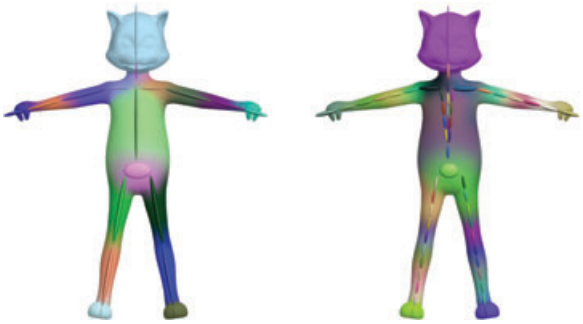

Figure 11: *Skinning weight distribution: on the left are the original weights; on the right is the new weight distribution in the sub-joint hierarchy after exaggeration. The color of each vertex of the mesh is computed by weighted blending of the color of joints which are bound to the vertex.*

Suppose an original joint J_k is bound to a skin mesh vertex v, with weight $w_{v,k}$, and suppose that the joint has been subdivided into the sub-joints $J'_{k,l}$, $1 \leq l \leq m_k$ during exaggeration. The original weight $w_{v,k}$ is redistributed to the new weights $w'_{v,k,l}$, in proportion to the inverse of the distance between the mesh vertex v and the line segment of the link connecting the sub-joints. Figure 11 shows the redistribution of weights in a cat character.

7. Results

We tested the effectiveness of our algorithm on several sequences of motion capture data. For a full appreciation, we recommend watching the associated video material. The middle row of Figure 12 shows the results of positioning the RJGs without blending the simulated motion. We can see that the original motion capture data (the first row in Figure 12) has clearly been exaggerated in a rubber-like manner. The

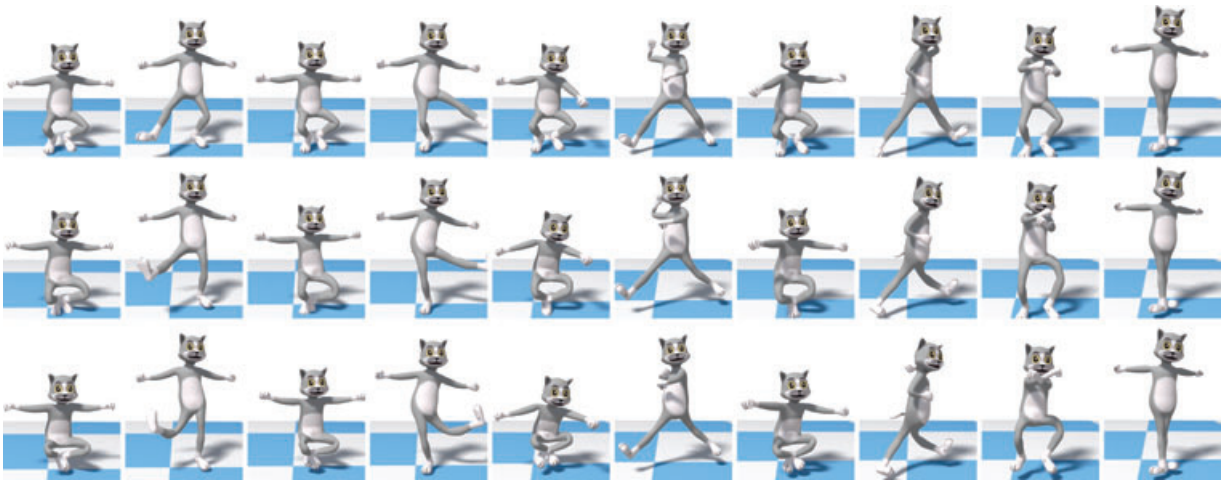

Figure 12: *The Cossack dance motion of this cat character is effectively exaggerated by our method. Top row: original motion capture data. Middle row: applying rubber-joint group positioning. Bottom row: applying mass-spring simulation and blending.*

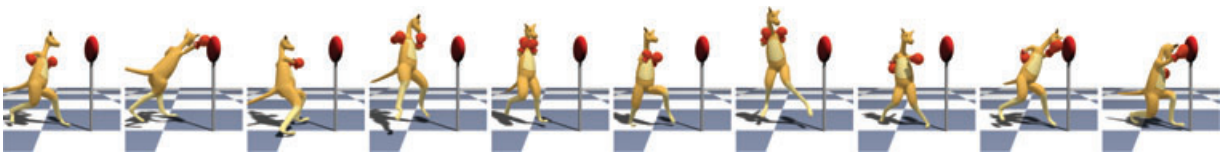

Figure 13: *The boxing motion of a kangaroo character. Forelegs (front limbs) of the character are nicely stretched to hit the target.*

bottom row of images in the same figure shows the results of blending simulated follow-through with the exaggerated motion. Physically plausible follow-through motions make the resulting motion more dynamic and interesting, and more cartoon-like. Note that the knee bending back of the cat in the third row of Figure 12 seems to be undesired artifacts; however, it can be an interesting effect which is called 'breaking the joint' [Wil01] by artists. Figure 13 is an example of a boxing motion that is first exaggerated then post-processed by the method described in Section 6. We can observe that the boxing kangaroo character is nicely exaggerated while it keeps hitting the red target by our method.

Figure 14 compares our method with the other motion exaggeration techniques. The motions in Figure 14(b) [LS01] and (c) [WDAC06] are exaggerated by rotating the joints excessively. Our results in Figure 14(e) show squashed or stretched links that cannot be achieved by these other techniques.

The results in Figure 14(d) [WDAC06] do contain links of modified length (even though Wang *et al*. do not explicitly mention the exaggeration of joint trajectories). However, the character body of our results seems to be made of rubber, because of its bending effect generated by the sub-joint hierarchy and the RJG technique.

Table 1 gives the computation time per frame for each step in our algorithm for several examples. Running times were measured on a 2.13 GHz Intel Core2 CPU with 2 GB of memory. All the example motions have the same joint hierarchy with 23 joints, and all the sub-joint hierarchies have 58 joints. We can see that the time needed to generate one frame is short enough for our algorithm to be used in real time.

8. Conclusions and Future Work

We have proposed a novel method to convert a captured motion to a rubber-like exaggerated motion. We have resolutely broken the traditional link-length constraint in order to extend motions spatially. We also refine the original joint hierarchy into a new sub-joint hierarchy, defining the position of the sub-joints using Bézier curve interpolation in order to emulate the bending effect of rubber. Finally, we apply a mass-spring simulation to the sub-joint hierarchy to generate physically plausible follow-through motions.

Our method has some limitations. First, the amount of data required for exaggerated motions is much greater than that required for the original motion because of the additional joints and the need for 6 DOF channel data. Second, because

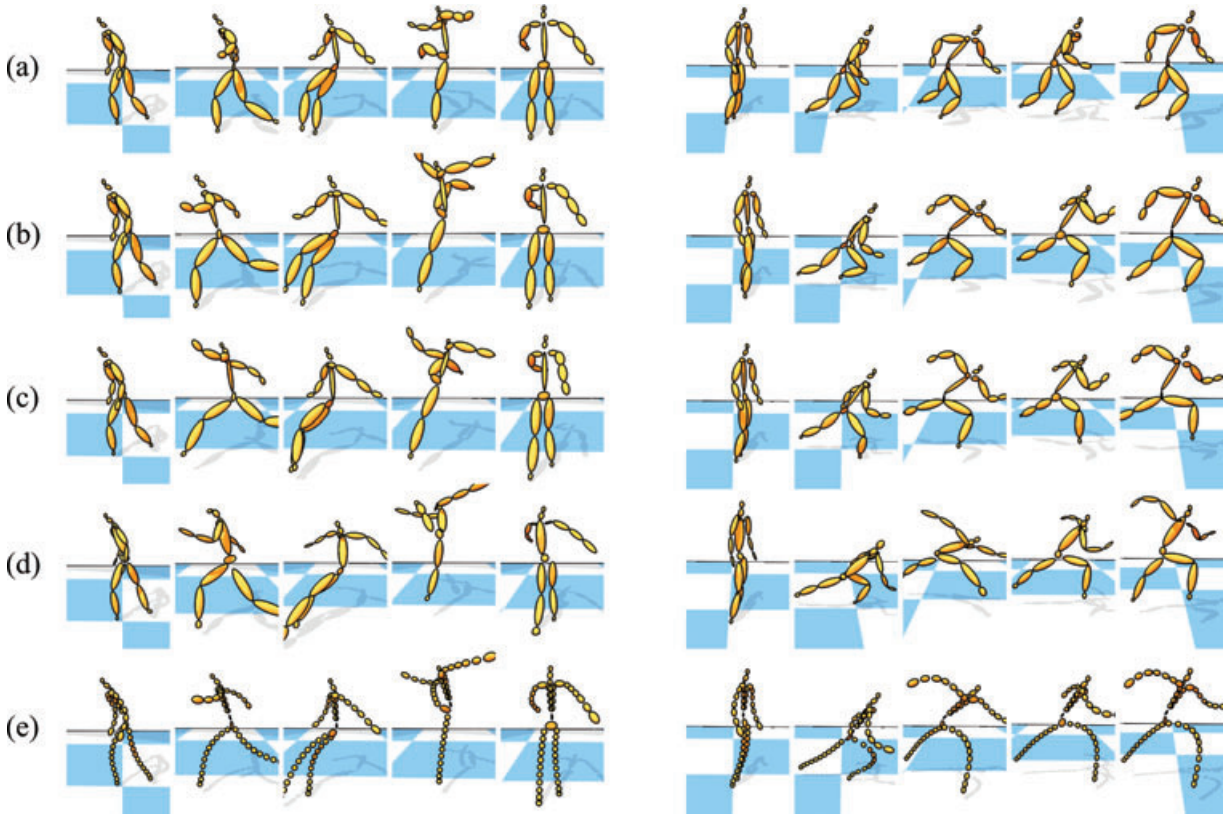

Figure 14: *Results for two motions, 'soccer kick' (left column) and 'running' (right column): (a) original motion; (b) Lee and Shin's technique [2001]; (c) Wang et al.'s technique [2006]; (d) the same technique with violation of link-length constraints; and (e) our results.*

Table 1: *Computation time per frame for each step in our algorithm (ms).*

	Curve extra-polation	Mass-spring simulation and blending	Total
Walk	1.35	10.41	11.76
Run	1.18	4.64	5.82
Kick	1.28	11.52	12.80
Cossack	1.29	10.42	11.71

our method changes the length of links, we cannot guarantee that the volume of the resulting mesh will be preserved. Finally, we do not consider a penetration between each link of a character; thus, careful control of exaggeration parameter is needed for generating collision-free and plausible results.

There are many potential improvements to our technique. First, we should consider automatic determination of the exaggeration parameters. We should consider a more convenient method for controlling the amount of exaggeration. Controlling the amount of exaggeration for specified partial bodies of the character should also be considered. Second, other physical simulation methods might be used instead of the simple mass-spring simulation, such as a mass-spring system with a more complicated structure, or the modal warping technique introduced by Choi and Ko [CK05]. Finally, we could apply a time-warping technique [WLvdP06] to an exaggerated motion in order to generate slow-in and slow-out effects [Las87].

Acknowledgments

This work was supported by grant No. R01-2004-000-10117-0 from the Basic Research Program of the Korea Science & Engineering Foundation. The data used in this project was obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217.

References

[ABC96] AMAYA K., BRUDERLIN A., CALVERT T.: Emotion from motion. In *Proceedings of the Graphics Interface 1996 Conference* (Toronto, Canada, Canada, 1996), Canadian Information Processing Society, pp. 222–229.

[BLCD02] BREGLER C., LOEB L., CHUANG E., DESHPANDE H.: Turning to the masters: Motion capturing cartoons. In *Proceedings of ACM SIGGRAPH 2002* (New York, 2002), ACM Press, pp. 399–407.

[BW95] BRUDERLIN A., WILLIAMS L.: Motion signal processing. In *Proceedings of ACM SIGGRAPH 1995* (1995), ACM Press, pp. 97–104.

[CK00] CHOI K.-J., KO H.-S.: Online motion retargeting. *The Journal of Visualization and Computer Animation 11* (2000), 223–235.

[CK05] CHOI M. G., KO H.-S.: Modal warping: Real-time simulation of large rotational deformation and manipulation. *IEEE Transactions on Visualization and Computer Graphics 11*, 1 (2005), 91–101.

[CKB04] CHAUDHURI P., KALRA P., BANERJEE S.: A system for view-dependent animation. *Computer Graphics Forum 23*, 3 (2004), 411–420.

[CPIS02] CHENNEY S., PINGEL M., IVERSON R., SZYMANSKI M.: Simulating cartoon style animation. In *Proceedings of the 2nd International Symposium on Non-photorealistic Animation and Rendering* (2002), ACM Press, pp. 133–138.

[CRE01] COHEN E., RIESENFELD R. F., ELBER G.: *Geometric Modeling with Splines: An Introduction*. A K Peters, 2001.

[dB06] DE JUAN C. N., BODENHEIMER B.: Re-using traditional animation: Methods for semi-automatic segmentation and inbetweening. In *Proceedings of the Eurographics/SIGGRAPH Symposium on Computer Animation* (Vienna, September 2006), pp. 223–232.

[Gle98] GLEICHER M.: Retargetting motion to new characters. In *Proceedings of ACM SIGGRAPH 1998* (1998), ACM Press, pp. 33–42.

[GMHP04] GROCHOW K., MARTIN S. L., HERTZMANN A., POPOVIĆ Z.: Style-based inverse kinematics. *ACM Transactions on Graphics 23*, 3 (2004), 522–531.

[HHD04] HALLER M., HANL C., DIEPHUIS J.: Nonphotorealistic rendering techniques for motion in computer games. *Computer Entertainment 2*, 4 (2004), 11–11.

[HPK*02] HILL H., POLLICK F. E., KAMACHI M., TROJE N. F., WASON T., JOHNSTON A.: Using the principles of facial caricature to exaggerate human motion. In *Proceedings of the European Conference of Visual Perception 2002* (2002).

[HPP05] HSU E., PULLI K., POPOVIĆ J.: Style translation for human motion. *ACM Transactions on Graphics 24*, 3 (2005), 1082–1089.

[HRvdP04] HARRISON J., RENSINK R. A., VAN DE PANNE M.: Obscuring length changes during animated motion. In *Proceedings of ACM SIGGRAPH 2004* (2004), ACM Press, pp. 569–573.

[KCSL06] KIM J.-H., CHOI J.-J., SHIN H. J., LEE I.-K.: Anticipation effect generation for character animation. In *Proceedings of the Computer Graphics International Conference* (2006), pp. 639–646.

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. In *Proceedings of ACM SIGGRAPH 2002* (2002), ACM Press, pp. 473–482.

[Las87] LASSETER J.: Principles of traditional animation applied to 3D computer animation. In *Proceedings of ACM SIGGRAPH 1987* (1987), ACM Press, pp. 35–44.

[LGXS03] LI Y., GLEICHER M., XU Y.-Q., SHUM H.-Y.: Stylizing motion with drawings. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, 2003), Eurographics Association, pp. 309–319.

[LHP05] LIU C. K., HERTZMANN A., POPOVIĆ Z.: Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics 24*, 3 (2005), 1071–1081.

[LS01] LEE J., SHIN S. Y.: A coordinate-invariant approach to multiresolution motion analysis. *Graphical Models 63*, 2 (2001), 87–105.

[MG03] MOHR A., GLEICHER M.: Building efficient, accurate character skins from examples. In *Proceedings of ACM SIGGRAPH 2003* (2003), ACM Press, pp. 562–568.

[MK05] MUKAI T., KURIYAMA S.: Geostatistical motion interpolation. *ACM Transactions on Graphics 24*, 3 (2005), 1062–1070.

[NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: Fibermesh: Designing freeform surfaces with 3D curves. ACM Press, p. 41.

[Rad99] RADEMACHER P.: View-dependent geometry. In *Proceedings of ACM SIGGRAPH 1999* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 439–446.

[Red84] REDMAN L.: *How To Draw Caricature*. Contemporary Books, 1984.

[Sho85] SHOEMAKE K.: Animating rotations with quaternion curves. In *Proceedings of ACM SIGGRAPH 1985* (1985), ACM Press, pp. 245–254.

[SLSG01] SHIN H. J., LEE J., SHIN S. Y., GLEICHER M.: Computer puppetry: An importance-based approach. *ACM Transactions on Graphics 20*, 2 (2001), 67–94.

[TK00] TAK S., KO H.-S.: Example guided inverse kinematics. In *Proceedings of the International Conference on Computer Graphics and Imaging* (2000), pp. 19–23.

[UAT95] UNUMA M., ANJYO K., TAKEUCHI R.: Fourier principles for emotion-based human figure animation. In *Proceedings of ACM SIGGRAPH 1995* (1995), ACM Press, pp. 91–96.

[vFTS07] VON FUNCK W., THEISEL H., SEIDEL H.-P.: Elastic secondary deformations by vector field integration. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2007), Eurographics Association, pp. 99–108.

[WB97] WITKIN A., BARAFF D.: Physically based modeling: Principles and practice. ACM SIGGRAPH 1997 Course Notes, 1997.

[WDAC06] WANG J., DRUCKER S. M., AGRAWALA M., COHEN M. F.: The cartoon animation filter. In *Proceedings of ACM SIGGRAPH 2006* (2006), ACM Press, pp. 1169–1173.

[Wil01] WILLIAMS R.: *The Animator's Survival Kit: A Manual of Methods, Principles and Formulas*. Faber and Faber, 2001.

[WLvdP06] WHITE D., LOKEN K., VAN DE PANNE M.: Slow in and slow out cartoon animation filter. SIGGRAPH 2006 Poster, 2006.