COMPUTER-AIDED DESIGN & APPLICATIONS, 2017
VOL. 14, NO. 2, 170–179
http://dx.doi.org/10.1080/16864360.2016.1223428

Interactive cartoon-style motion generation of avatar

Xiubo Liang

Zhejiang University, China

ABSTRACT
In 3D character animation, the expressiveness of the motions is very important for the immersion and the perception in a virtual environment. Traditional animation is often superior to show this expressiveness. However, the resulting animation is typically not interactive. We present in this paper an interactive and intuitive method to add expressiveness to avatars by the exaggeration of the characters and their motions. We first use the data from a sensor holding by the user to create a parameterized order. Then this intuitive order is interpreted by the system to drive the exaggeration of the involved limbs and the deformation of the mesh of the avatar. We finally provide a post-exaggeration module adapting the resulting motion to the environment and placing the camera for an optimal perception. Experimental results show that the method can efficiently convert user's interaction intent to expressive cartoon-style character animation.

KEYWORDS
Motion Recognition; Cartoon Effect; Motion Adaptation

1. Introduction

A lot of research has been done on creating realistic motions for character animation. These methods are excellent in generating realistic motions, but they are not suitable for directly producing expressive and non-photorealistic motions. On the other hand, traditional keyframe animation is widely used to animate such motions of virtual characters. But this involves difficult and time consuming work, even when used by a skillful animator. In comparison of non-living objects, animating a human character has some specific issues due to its skeletal structure and physiological system. Nevertheless, expressive and stylized human motions can be nicely represented by temporarily stretching, breaking link-length constraints, creating new sub-joints and more. These techniques produce great results, but their parameterization remains difficult to setup in real-time for interactive purposes.

In this paper, we present an interactive and intuitive way to give expressiveness to a human motion by adding exaggeration. Exaggeration is useful to help the audience to focus on a more important space-time sequence of a motion. Animators can so increase the perception on a specific point of interest. Exaggerated motions are not realistic but they represent the situation nicely, by giving more expressiveness and emotion. Our aim is to provide a novel exaggeration method involving intuitive user's gestures from an interactive device. Cartoon-style animations or video games are good applications for this kind of motion generation. There are three main phases in our interactive cartoon-style motion generation system. Firstly, the user performs an action by holding an accelerometer embedded in a Wiimote and our system will recognize the action with a pre-trained Hidden Markov Model (HMM) and turn it into an interaction command. Secondly, the recognized command is employed to drive the motion exaggeration and mesh deformation of the avatar to generate cartoon effects. Finally, the exaggerated character animation is adapted to fit the constraints of the virtual environment and the camera is placed in an optimal perception to give the best display to the user.

The primary contributions of this paper are as follows:

- We present a novel approach to interact with the virtual characters with a low-cost accelerometer while avoiding the boring work of collecting lots of training samples.
- The motion and body of the virtual character are both exaggerated automatically according to user's performance to obtain the interesting cartoon effects.
- The exaggerated motion of the virtual character and the view point of the virtual camera are further adapted to give the best visual effect to the user.

The remainder of this paper is organized as follows. We first review the related work in Section 2, and explain

CONTACT Xiubo Liang xiubo@zju.edu.cn

the architecture and pipeline of our cartoon-style motion generation method in Section 3. Then, we describe how to acquire and analyze the user-performed actions with a handheld accelerometer in Section 4. Section 5 presents the implementation details for motion exaggeration and mesh deformation. In Section 6, we give the adaptation methods to meet the constraints in the virtual environment. The experimental results and a brief discussion of our approach are given in Section 7. Finally, we give the conclusion and outlook of this paper in Section 8.

2. Related work

Traditional human computer interfaces are not intuitive enough for the control of avatars. In recent years, new input devices (e.g. Wiiremote and Kinect) are used to implement the virtual human control interface based on the somatosensory interaction, which makes full use of the human perception ability to express the control intention in a natural way [16, 15]. Such interactive systems only collect part-body motions with a few motion sensors or cameras, and then reconstruct the full-body motions based on a pre-captured motion database [19, 7]. A key problem in such systems is action recognition which refers to the process that computers cognizing user's motion state by certain means over a period of time. The process of action recognition generally comprises two aspects: firstly extracting the features of the motion, and secondly recognizing actions using the extracted motion features. Sanna et al. solves the problem of user-independent gesture recognition based on a preprocessing approach of tilt compensation and normalization [14]. Lv and Nevatia present an algorithm to automatically segment and recognize basic human actions by decomposing the high dimensional 3D joint space into a set of low dimensional feature spaces [10]. Cao and Balakrishnan present a hybrid recognition and prediction engine which can significantly improve gesture recognition performance and reduce users' effort of making the gestures before achieving good results [1].

Fundamental principles of traditional animation present bones as geometries which can be squashed and stretched, and joints can be broken if it makes for a more expressive motion [20]. The limbs of characters are often enlarged to accentuate a motion or imply an emotion. Majkowska and Faloutsos proposed motion editing operations to create complex super-human motions or acrobatic stunts that would be difficult to record in a motion capture studio [11]. Kwon and Lee simulate rubber-like exaggerations in order to convert motion capture data into cartoon-like movements [9]. Many techniques that follow traditional animation principles simulate squash and stretch, which is the most significant shape distortion in exaggerated motions [6, 2]. Wang et al. have applied the Laplacian of a Gaussian filter to general animation signals to generate anticipation and follow-through effects [17]. Noble and Tang present a tool that takes skeleton-driven animations and generates expressive deformations to the character geometry which is warped along automatically computed limbs's line of motion [12]. After edited by some motion editing algorithm, the resulting motion should be modified to fit the constraints in the virtual environment. Recently, more and more researchers introduced the physical constraints into their work of motion retrieval and synthesis [18, 3]. In order to satisfy the visual reasonableness, such systems should be able to generate motions which meet some physical laws [4,5,21].

3. Overview

The main objective of our system is to give more expressiveness to the motion in a natural way. If we know on what part of the body the user wants to interact, the global expressiveness will be improved by exaggerating this part. We present a method using an interactive data, a gesture from the user, as the main parameter defining the exaggeration. We studied the possibility of making the deformation dependent on one other motion than the original which can describe the way to deform it. The main process is shown in Figure 1. Two inputs are used: the current motion of the avatar and a gesture performed by the user. The user observes the character moving within the virtual environment. At any moment, he can decide to see this character exaggerated. Depending on the user's order, the movement of a body part is then exaggerated. The modifications are calculated from the way to order the action. We use an acceleration sensor to capture the properties of the gesture, and interpret these data to proceed to the exaggeration. The process can be applied on multiple applications, and especially on game-like scenarios.

The major algorithmic steps of the interactive cartoon-like motion generation are explained below:

1. Gesture analysis. The user perform an interactive action by holding a Wiimote in his hand. The motion recognition module recognize the action with an HMM which is trained by the samples pre-performed by the user and the automatically generated samples by adding noise. The performed action is also be employed to calculate the quantitative measurement of magnitude and duration.
2. Exaggeration. The corresponding part of the virtual character is exaggerated by joint position modification and mesh deformation. For motion

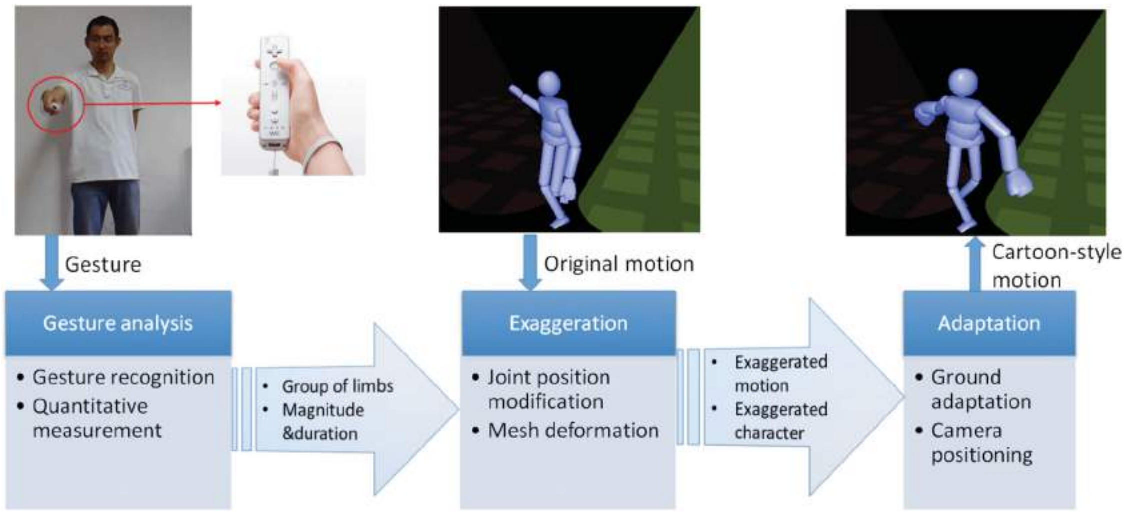

Figure 1. Overview of the interactive cartoon-style motion generation.

exaggeration, we modify the velocity curve of each joint in the three-dimensional space. For mesh exaggeration, we propose to modify the width and the height of the meshes to enlarge the limbs relatively to the hierarchy of the joints.

3. Adaptation. The exaggerated motion is further modified to fit the constraints in the virtual environment. We use the ground constraints to avoid the penetration of limbs into the ground and compute the optimal position of the camera to display the generated cartoon effects of the avatar.

4. Gesture analysis

We get the acceleration data of a motion from the Wiimote held by the user. The sensed data are used for gesture recognition and quantitative measurement.

4.1. Gesture recognition

Our recognition algorithm includes three major steps: sampling, preprocessing, training and recognizing. To achieve good recognition accuracy, there must be enough training samples. Kay pointed that the detectability can be improved by adding noise [8]. So we reduce the users' labor by generating samples automatically. Users only need to perform a gesture one time, the system will generate the left samples by adding noise. There are two random noise distributions used in our system: Uniform distribution and Gaussian distribution. We achieve the best noise level by experiments according to the *signal to noise ratio (SNR)* which is the ratio of signal variance to noise variance.

Performing the same gesture, the users will certainly produce different movements generating different accelerations. This leads to the multiplicity of the gestures. The length of the samples can be very different, which increases the difficulty of the recognition. We solve this problem by a method of *interpolation and resampling*: for each acceleration sample, we interpolate their values between two sample points and get a cubic spline. Then, we resample the spline to the same length. And because of the multiplicity of the gestures, the main feature of a gesture is submerged in redundant information. We employ a principle component analysis (PCA) to extract and rearrange the features. The feature sequences are then introduced to a Hidden Markov Model (HMM) for training and recognizing.

In our system, each HMM has three hidden states and each state contains a 3-component mixture of Gaussian. Supposing that we have M kinds of gestures and each of which has a corresponding HMM (represented by λ_i), forming a M-length vector of HMMs. Assuming that after preprocessing, the sample is represented by a feature sequence $O = (o_1 o_2 \ldots o_T)$. Once λ_i is learned, the probability of the feature sequence O is computed using the *Forward* algorithm [13]. The gesture with the maximum probability in the vector is selected. The main steps of training and recognizing are then as follows:

1. Estimate the parameters of each HMM with the *Baum-Welch* algorithm [13].
2. Compute the probability $P(O|\lambda_i)$ with the *Forward* algorithm.
3. Find the gesture with the maximum $P(O|\lambda_i)$ using the following *formula (1)*:

$$\mathit{Gesture}(\mathrm{O}) = \arg\max[P(O/\lambda_i)], \quad i = 0 \ldots M \quad (1)$$

Both *Forward* and *Baum-Welch* algorithms need to compute $P(o_t|i)$, the probability of observing o_t given the state i at time t. The continuous HMM uses a Probability Density Function (PDF) to estimate $P(o_t|i)$ because there is no point probability in continuous distribution. In our approach, we use the Gaussian mixture model, and $P(o_t|i)$ is computed as follows:

$$P(o_t|i) = \sum_{m=1}^{3} c_{im} \frac{1}{(2\pi)^{\frac{D}{2}}|U_{im}|^{\frac{1}{2}}} e^{-\frac{1}{2}(o_t-\mu_{im})U_{im}^{-1}(o_t-\mu_{im})^T} \tag{2}$$

where c_{im} is the coefficient, μ_{im} is the mean vector and U_{im} is the covariance matrix for the mth mixture component in state i.

4.2. Quantitative measurement

We use two quantitative parameters: the magnitude and the duration. The magnitude is the 'size of the order'. The larger movement the user performs, the more exaggerated the resulting motion will be. The duration represents the time of the exaggeration. The longer the input order takes, the longer the resulting motion exaggeration will be.

The sensed data is a collection of acceleration at each frame, a button on the Wiimote is used to control the beginning and the end of the order which is an intuitive way to delimit the order. The magnitude of the gesture is calculated by a double integration of the accelerations. The mapping between the space displacements of the gesture and the magnitude of the exaggeration is scaled into usable values. The duration can be directly set by multiplying the frame rate and frame count.

5. Exaggeration

The inputs to the exaggeration module are the current motion which will be modified, the group of limbs involved according to the user's order and the magnitude and the duration of the exaggeration. Both the motion and the character are modified to achieve the desired effect.

5.1. Motion exaggeration

We propose to consider a way to maintain the style and the features of the motion. To stretch the limbs, our method does not directly modify the trajectories of the joints, but the velocities. Direct exaggerations on the trajectories of the joints could sometimes change the original style of the motion and does not have a good robustness. So we exaggerate the velocity curve of each joint in the three-dimensional space to get more respectful and smooth result.

To get the velocity curve, we use a cubic spline curve to fit the trajectory of a joint. Then, we get the differential coefficients of the curve, which are the velocities of the joint. We exaggerate these velocity data, and the final resulting motion is obtained from the integral of the exaggerated velocity curve.

There are several methods to exaggerate the velocity trajectory. We propose one that can carry out the effect of fade in and fade out. This mechanism allows a smooth animation between the original and the exaggerated motions both at the beginning and at the end of the modification. Our algorithm also shows the follow-through effect, commonly observed in cartoon style animations.

Let $v_{ko}(t)$ denote the original velocity curve of the kth joint J_k, with the time parameter ($t \in [0, T]$). An exaggerated version $v_{ke}(t)$ of $v_{ko}(t)$ can be computed as follows:

$$v_{ke}(t) = F(t)v_{ko}(t) \tag{3}$$

F(t) is the function used to exaggerate the original velocity, described by the following equation:

$$\int_0^T F(t)v_{ko}(t)dt = A \tag{4}$$

where A is the displacement between the start position and the end position. Many kinds of functions could fit this equation, we choose the quadratic function $F(t) = at^2 + bt + c$ that gives good results regarding to our experiments. The coefficients *a, b* and *c* are computed as follows:

$$\begin{aligned} &\int_0^T (at^2 + bt + c)v_{ko}(t)dt \\ &\quad = a\int_0^T t^2 v_{ko}(t)dt + b\int_0^T t v_{ko}(t)dt + c\int_0^T v_{ko}(t)dt \\ &\quad = \mathrm{a}[t^2 s_{k0}(t)|_0^T - 2\int_0^T t s_{ko}(t)dt] \\ &\qquad + b[t s_{ko}(t)|_0^T - \int_0^T s_{ko}(t)dt] + c s_{ko}(t)|_0^T = A \end{aligned} \tag{5}$$

$$s_{ko}(t) = \int_0^T v_{ko}(t)dt, \quad c = F_0, \frac{4ac - b^2}{4a} = B \tag{6}$$

F_0 is the constant value of *F(t)* at $t = 0$. If we set $F_0 = 1$, the velocity curve will so be exaggerated from its original value *v(0)*. B is also a constant value we need to set. It corresponds to the maximum of the quadratic function *F(t)*. In our application, this value is the magnitude determined by the quantitative measurement module of the user's gesture analysis. $s_{ko}(t)$ is the trajectory of the

kth joint j_k which we want to compute. And we can pose. $C = \int_0^T s_{ko} dt$ and $D = \int_0^T ts_{ko}(t)dt$. Let's denote. $tempB = \frac{-4(T_{s_{ko}}(T)-F_0)(B-F_0)}{T^2(s_{ko}(T)-2D)}$ and $tempC = \frac{4(1-F_0)A(B-F_0)}{T^2(s_{ko}(T)-2D)}$, then we obtain the searched coefficients by:

$$a = \frac{(1-F_0)A - (ts_{ko}(T)-c)b}{t^2(s_{k0}(T)-2D)},$$
$$b = \frac{-tempB + \sqrt{tempB^2 - 4tempC}}{2}, \quad c = F_0 \qquad (7)$$

Finally, we can multiply the original velocity curve with *F(t)* to get the exaggerated motion. One result is shown in Figure 2. The limbs first stretch slowly from their original sizes. Then, they are modified according to the maximum magnitude and the original motion. And finally, they return to their original sizes at the end of the exaggeration.

5.2. Mesh deformation

A consequence of the exaggeration of the position of the joints is the distortion of the sizes of the limbs (as seen in Figure 2). To keep the geometry consistency of the character, we so need to modify the shape of the limbs. Moreover, we also use this modification to increase the perception of the exaggeration.

As the adaptation of the shape depends on the modeling techniques of the representation of the virtual character, we present in this section a method applied on one kind of representation: one mesh for one limb. The virtual character is composed of 16 segments representing: the head, the arms and forearms, the hands, the thighs, the lower legs, the feet, and the trunk (made of three parts). These meshes are described and rendered using the positions and the normals of each vertex. By modifying the positions of theses vertices, we can modify the shape of the character.

Let's suppose the local coordinate system *(x, y, z)* of each mesh M: x the width of the limb, y the length and z the height. First, for each frame of the exaggeration, the y component of each vertex v of M is modified to fit the new length of the limb. We use here a property of our meshes: the y component of the normals of the vertices N_v are positive in the proximal half of the mesh and negative in the distal half. We so translate the distal half by $M_{newLength} - M_{originalLength}$, where these lengths are computed from the positions of the proximal and distal joints of the corresponding limb.

This deformation only depends on the joints exaggeration, but we also want to add expressiveness to the motion thanks to the shape of the character. We propose to modify the width and the height of the meshes. Our idea is to enlarge the limbs relatively to the hierarchy of the joints. The deeper in the hierarchy the proximal joint stands, the larger the associated mesh will be. To describe this evolution, we define a value, *Mesh Scale* (*MS*), which depends on the magnitude of the user's order and the index i of the current mesh in the selected group hierarchy:

$$MS_i = magnitude + i * F \qquad (8)$$

where F is a constant coefficient, called the *Expansion Factor*. The value of this factor and its effect on the exaggeration are discussed in the experiments section. i takes

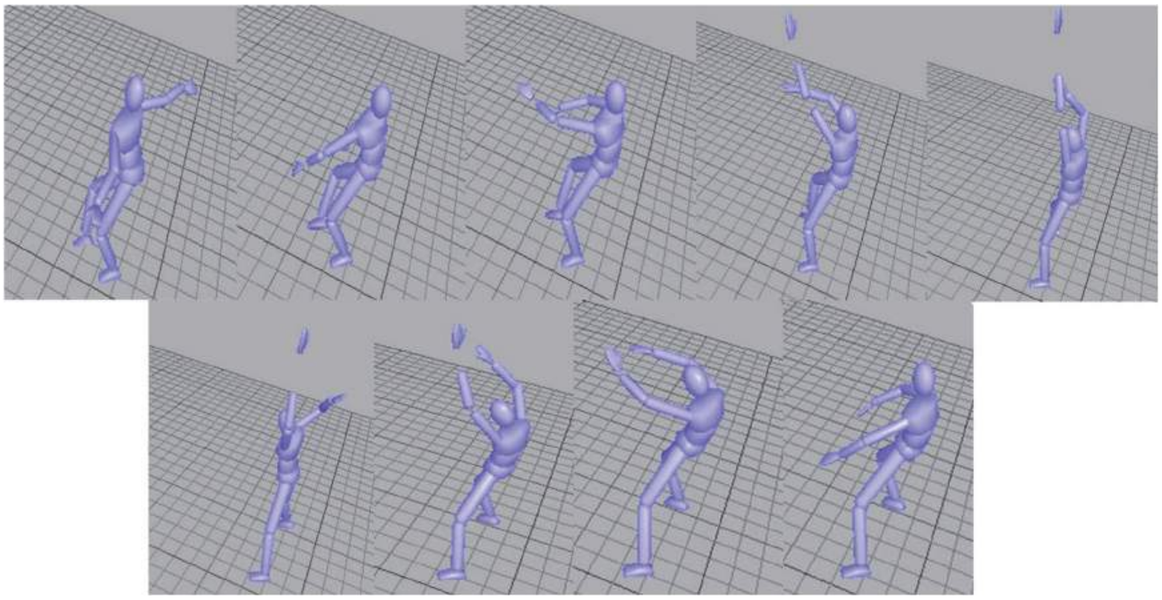
Figure 2. A result of a left arm exaggeration (without modification of the character).

its value from 0, for the root limb of the selected group, to the number of limbs−1 for the last of the group.

In order to conserve the ratio between x and z, MS represents the scale on the width and the scale on the height of each M. We use the normal of the vertices on these axis as the direction of the deformation (see Figure 3). The positions of the vertices are then updated to:

$$v_{(x,z)} = v_{(x,z)} + N_{v_{(x,z)}} \times MS_M \times \frac{M_{newLength}}{M_{originalLength}} \quad (9)$$

The deformation is scaled by the ratio between the new size and the original size of the limb in order to respect the original aspect. Moreover, MS_M is also interpolated using the fade in and fade out mechanism already used in the joints positions exaggeration. Some results of the shape modification are showed in Figure 4.

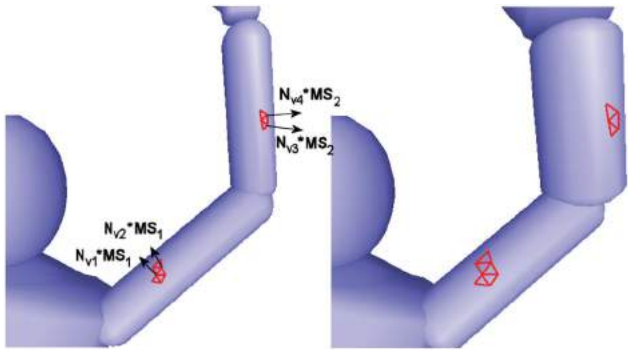

Figure 3. The mesh is deformed along the normal of each vertex N v. The scales MSM are described by a function which depends on the magnitude and the hierarchy of the limbs.

6. Adaptation to the environment

Virtual character animation implicitly exhibits constraints such as the fact that the feet do not (generally) slide when in contact with the floor. We present now how we deal with two kinds of these constraints: the penetration of limbs into the ground and the automatic positioning of the camera.

6.1. Ground constraint

After exaggeration, the sizes of the exaggerated limbs are longer than the original ones, and some joints may penetrate the ground. We need to guarantee that all the joints are not under the ground. We separate the ground constraint correction into two cases.

- *The lower body is involved in the exaggeration*: The positions of the feet can be under the floor and the support phase can have changed. To solve this problem, we used precomputed support phases and support joints. Before the beginning of the simulation, we compute the support phases thanks to an algorithm using the positions and the velocities of two potential contact joints: the toes and the ankles. In this algorithm, a foot is supposed in support if the position and the velocity of at least one of the joints are respectively under a predefined threshold. The support joint is then the lower joint of the support foot. After the exaggeration of the lower body, we determine if the new lower joint is the same as the original support joint at the specified frame. If it is, we get the vertical distance between the support joint and the ground, and rise up the character using this height. On the other hand, if the new lower joint is not the original

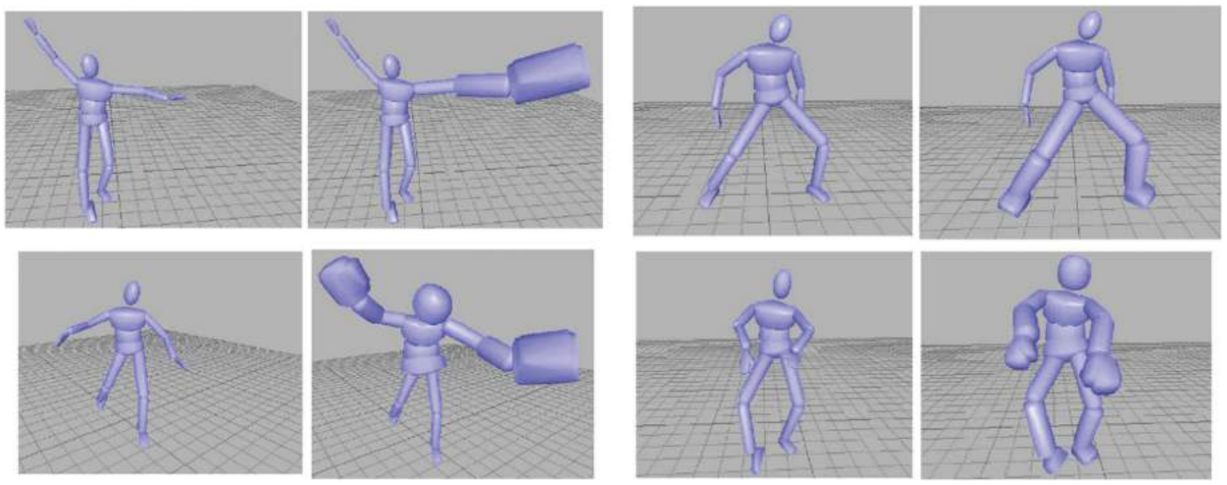

Figure 4. Four examples of the mesh deformation on: the left arm, the legs, the upper body and the full body. Results obtained with a magnitude of 3 and an expansion factor of 2.

support joint, we reduce the sizes of the limbs to obtain the same height as the support joint. And finally, rise up the character as described before (see Figure 5).

- *The lower body is not involved in the exaggeration*: In this case, our algorithm is simpler. We do not need to change the global position of the character, but only to correct the potential penetration of the upper joints. At each frame, we first detect the collisions of the exaggerated joints with the ground.

For these joints, we get the associated limbs, and we reduce their sizes to avoid the penetration. Let J_{under} denote a joint which is under the ground, and J_{above} denote the first joint which is above the ground in the same hierarchy. The scales applied on each limbs of the involved group are computed as

$$scale = \frac{J_{above}}{J_{above} - J_{under}}.$$

6.2. Camera positioning

Controlling the camera is an important issue for the perception of the 3D motions. Viewers can have completely different sensations of a same motion only because of the position of the viewpoint. Here, as we want to improve the perception of the displacements of the exaggerated limbs, we propose a method to compute the optimal position of the camera displaying these displacements. Let $J_{n,t}$ denote the position of the joint n in the hierarchy of the selected group of limbs at frame t. We can define the successive positions of an exaggerated joint by a set $J_{n,[0,T]}$, and the positions of each joints in the group at t can be represented by a set $J_{[1,N],t}$. To determine the orientation of the camera, we use the cross products defined by the following vectors (see Figure 6):

$$U_{n,t} = \overrightarrow{J_{n,t}J_{n+1,t}} \times \overrightarrow{J_{n,t}J_{n+1,t+1}}, \; V_{n,t} = \overrightarrow{J_{n,t}J_{n+1,t+1}} \times \overrightarrow{J_{n,t}J_{n,t+1}} \quad (10)$$

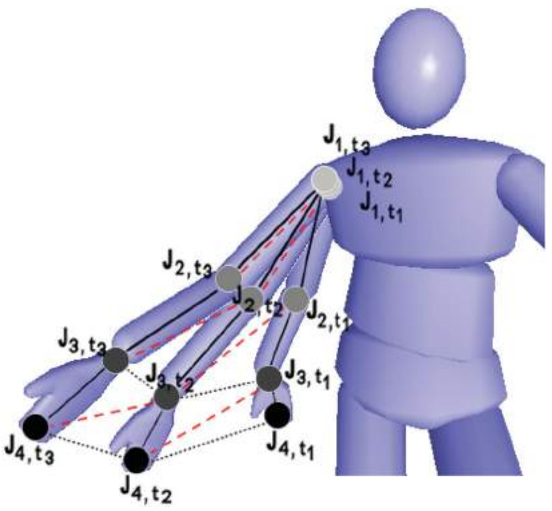

Figure 6. The kinematical data from the right arm: J_1, the shoulder, to J_4, the hand, are used to compute the average normal and position from time t_1 to t_3.

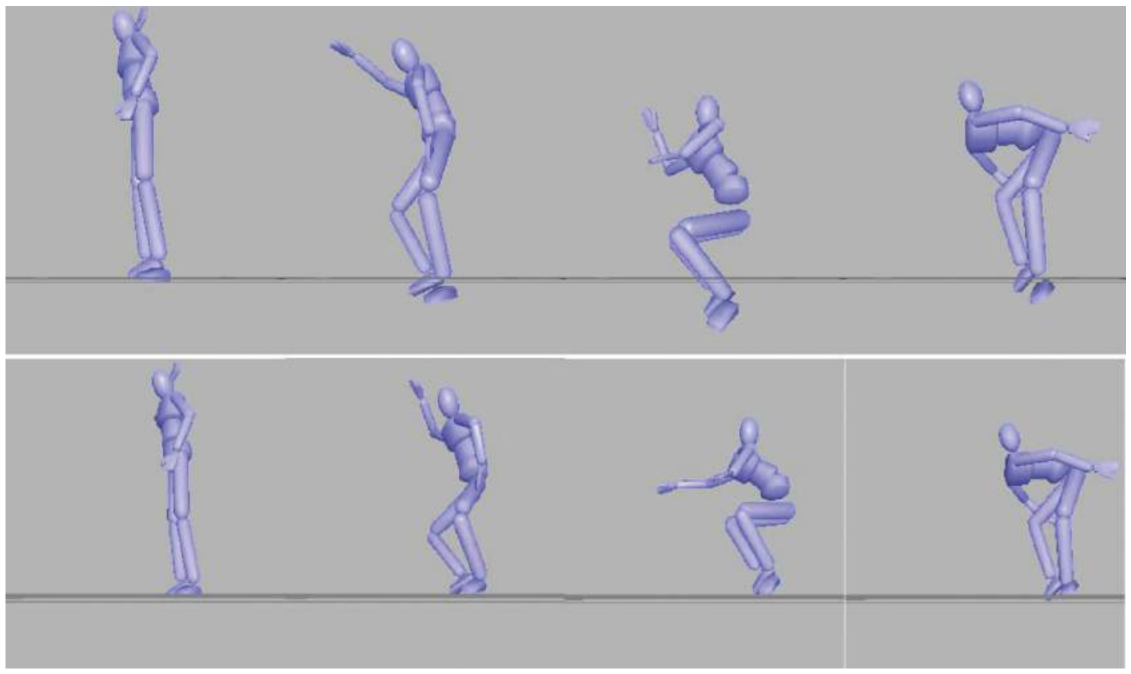

Figure 5. An example ground constraint correction on the lower body. (Left) An exaggerated motion without ground constraint correction. (Right) The same exaggerated motion with ground constraint correction.

Then, for all the frames $t \in [1, T]$ and all the joints $n \in [1, N]$ of the exaggeration, we compute the sum of the normalized U and V:

$$S = \sum\nolimits_{t=1}^{T} \sum\nolimits_{n=1}^{N} \frac{U_{n,t}}{\|U_{n,t}\|} + \frac{V_{n,t}}{\|V_{n,t}\|} \qquad (11)$$

By normalizing U and V, they are now independents of the sizes of the limbs, enabling a homogeneous weight. By the way, we can notice that a size-based weight method can be defined just by removing the U and V normalizations. The direction of the point of view D is then given by: $D = -\frac{S}{S}$.

We now have to define the center of view, i.e. the 3D position where the camera looks at. In this purpose, we use the average position of one exaggerated joint. We define, before the simulation, which joint will be the center of view n_G of each possible group of limbs. For example, the elbow joint is used as the center of the arm group. The global center of view CV is then defined by:

$$CV = \frac{1}{T} \sum_{t=1}^{T} J_{n_G,t} \qquad (12)$$

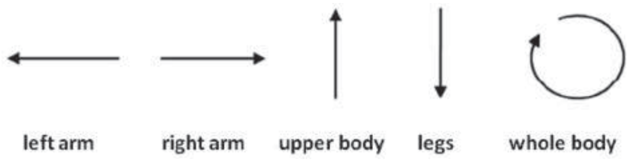

Figure 7. The input gestures used to select the exaggerated limbs.

The last parameter to set is the distance between the center of view and the position of the camera along the direction D. This parameter is a constant and can be defined by the user.

7. Experimental results

In this paper, a group of limbs is selected among the preset groups thanks to the gesture order (see Figure 7). We did a series of experiments on the recognition accuracy under various *SNR*. It shows that results are better (over 96%) when *SNR* is about 4. We also did some experiments on the flexibility of the method. The average recognition accuracy is over 95% for various magnitudes of the same gesture.

The *Expansion Factor F* of Equation (8) have been presented as a constant parameter. As our goal is to provide an automatic system, we study the influence of this parameter and exhibit a good value. Figure 8 shows examples of meshes deformations regarding to the magnitude and the factor F. We can observe that the magnitude A modifies the global sizes of the limbs while F modifies their relative sizes. After more experiments, we can assume that the best looking motions are obtained when $F \in [1.2,3.3]$. The initial value is then set to 2.0.

We tested the effectiveness of the camera positioning on many sequences of our scenario. Figure 9 shows some snapshots captured by the camera positioned by our algorithm. We can observe that our algorithm always well displays the motion of the exaggerated limbs whereas classical directions (up, front or side) may be inappropriate.

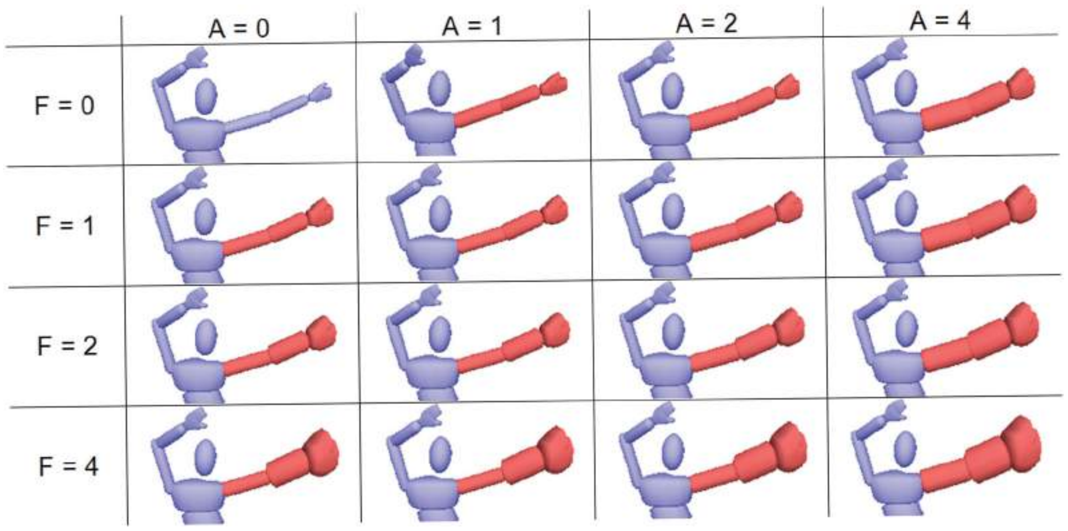

Figure 8. Evolution of the mesh deformation, regarding to the magnitude (A) and the expansion factor (F), without modifications of the joints positions.

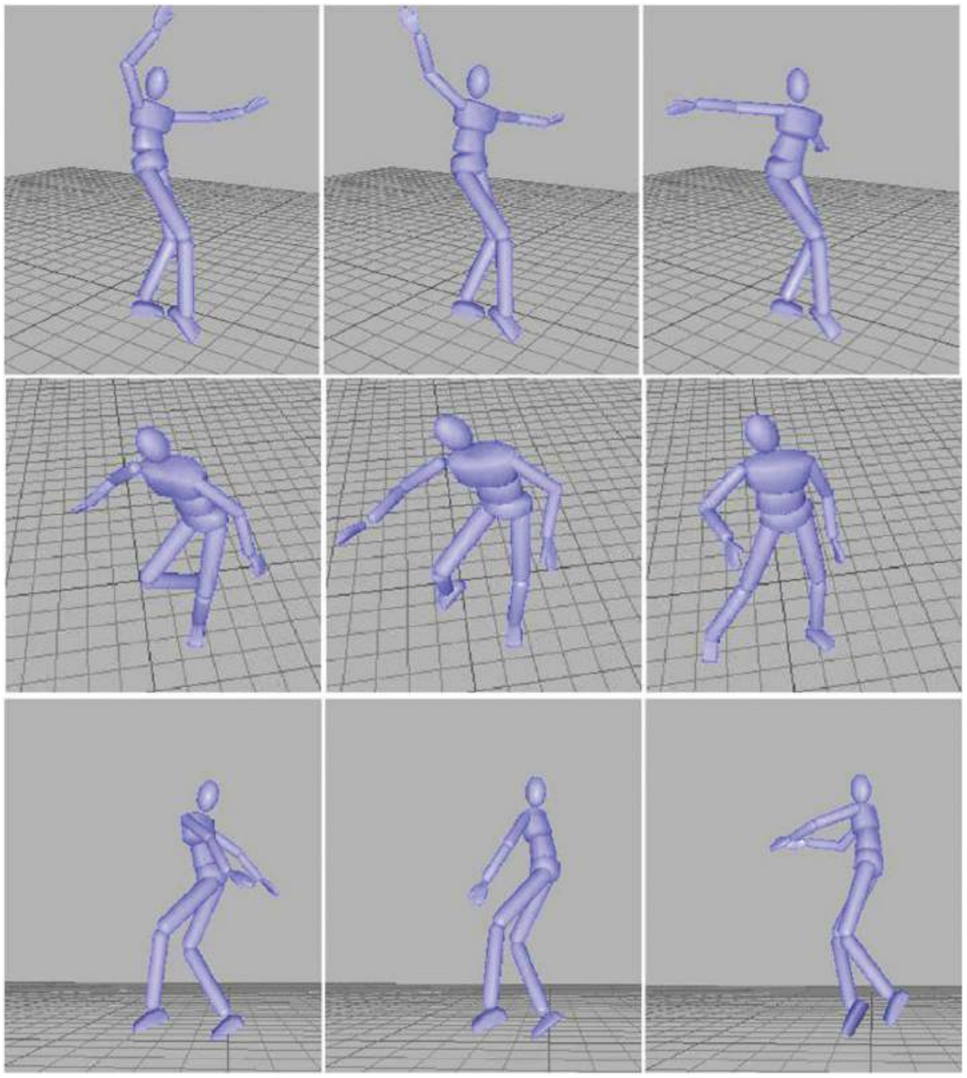

Figure 9. Three examples of camera positioning where the exaggerated group are: (left) the right arm, (middle) the upper body and (right) the full body.

8. Conclusion

This paper has presented an intuitive way to add expressive deformations to an animated character. The part of the body and the magnitude of the exaggeration are derived from the user's gesture. The new positions of the joints depend also on the original motion data. The mesh deformation and the positioning of the camera use these positions to improve the impression of exaggeration. As such, we add a layer of expressiveness to the resulting motion and help accentuate the underlying animation.

Many fields of applications can take advantage of our approach. As the process can also be used offline, applications such as motion editing for cartoon-like animations are particularly suitable. Moreover, the input parameters may be defined by other intuitive devices such as dance pads or full body sensors. And of course, multiple characters can interact through exaggerations. We can imagine fighting scenarios where an enlarged hit leads to an exaggerated falling motion of the opposite player.

Acknowledgements

The authors would like to thank the Ningbo Natural Science Foundation (Grant no: 2013A610064) and intelligence industry talent base of Ningbo for financial support.

ORCID

Xiubo Liang http://orcid.org/0000-0002-3605-3521

References

[1] Cao, X.; Balakrishnan, R.: Evaluation of an on-line adaptive gesture interface with command prediction. In Proceedings of Graphics Interface, pages 187–194, 2005.

[2] 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, page 133138, 2002.

[3] Geijtenbeek, T.; Van de panne, M.; Van der stappen, A.: Flexible muscle-based locomotion for bipedal creatures. ACM Transactions on Graphics, 2013, 32(6): 1–11

[4] Hämäläinen, P.; Eriksson, S.; Tanskanen, E.; Kyrki, V.; Lehtinen, J.: Online Motion Synthesis Using Sequential Monte Carlo. ACM Transactions on Graphics, 33(4), 2014.

[5] Hahn, F.; Martin, S.; Thomaszewski, Gross, M.: Rig-space physics. ACM Transactions on Graphics, 31(4): 1–8, 2012.

[6] Haller, M.; Hanl, C.; Diephuis, J.: Non-photorealistic rendering techniques for motion in computer games. Computer Entertainment, 2(4):11–17, 2004.

[7] Helten, T.; Muller, M.; Seidel, H. P.; Theobalt, C.: Real-time body tracking with one depth camera and inertial sensors. In Proc. IEEE Int. Conf. Comput. Vis., pages 1105–1112. IEEE, Dec. 2013.

[8] Kay, S.: Can detectability be improved by adding noise. IEEE Signal Processing Letters, 7:8–10, 2000.

[9] Kwon, J.-Y.; Lee, I.-K.: Rubber-like exaggeration for character animation. In Proceedings of the 15th Pacific Conference on Computer Graphics and Applications, page 1826, 2007.

[10] Lv, F.; Nevatia, R.: Recognition and segmentation of 3-d human action using hmm and multi-class adaboost. ECCV, 3954:359–372, 2006.

[11] Majkowska, A.; Faloutsos, P.: Flipping with physics: motion editing for acrobatics. In Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, page 3544, 2007.

[12] Noble, P.; Tang, W.: Automatic expressive deformations for implying and stylizing motion. The Visual Computer, 23(7):523–533, 2007.

[13] Rabiner, L.: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2): 257–286, 1989.

[14] Sanna, K.; Juha, K.; Panu, K.; Jani, M.: User independent gesture interaction for small handheld devices. IJPRAI, 20(4):505–524, 2006.

[15] Shum, H.; Ho, E.; Jiang, Y.; Takagi, S.: Real-time posture reconstruction for Microsoft Kinect. IEEE Transactions on Cybernetics, 43(5): 1357–1369, 2013.

[16] Tautges, J.; Zinke, A.; Kruger, B.; Baumann J.; Weber, A.; Helten, T.; Muller, M.; Seidel, H.-P.; Eberhardt, B.: Motion reconstruction using sparse accelerometer data, May 2011.

[17] Wang, J.; Drucker, S.; Agrawala, M.; Cohen, M.: The cartoon animation filter. In Proceedings of ACM SIGGRAPH'06, pages 1169–1173, 2006.

[18] Wang, J.; Hamner, S.; Delp, S.: Optimizing Locomotion Controllers Using Biologically-Based Actuators and Objectives. ACM Transactions on Graphics, 31(4): 25:1–25:11, 2012.

[19] Wei, X.; Zhang, P.; Chai, J.: Accurate realtime full-body motion capture using a single depth camera. ACM Trans. Graph., 31(6):188:1–188:12, Nov. 2012.

[20] Williams, R.: The Animator's Survival Kit. Faber and Faber, 2001.

[21] Zhao, W.; Zhang, J.; Min, J.; Chai, J.: Robust realtime physics-based motion control for human grasping. ACM Transactions on Graphics, 32(6): 1–12, 2013.