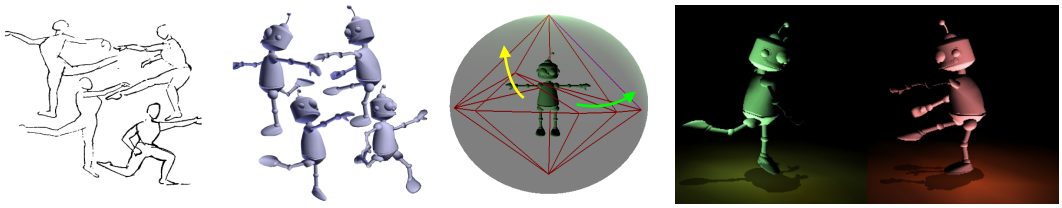
# Stylistic Reuse of View-Dependent Animations

Parag Chaudhuri Ashwani Jindal Prem Kalra Subhashis Banerjee
Department of Computer Science and Engineering,
Indian Institute of Technology Delhi, Hauz Khas, New Delhi 110016
{parag, ashwani, pkalra, suban}@cse.iitd.ernet.in




We take a set of sketches (a), reconstruct their poses and get the view-dependent models (b), then we create the view-sphere and move different cameras on it (c), giving view-dependent variations of an animation which we reuse to get the final stylized animation (d).

## Abstract

*We present in this paper a technique for stylized reuse of view-dependent animation. We illustrate with an example, how can a unique animation be synthesized by reusing the view-dependent variations of the same character. We start with a sequence of sketches given by an artist, create the view-dependent model corresponding to the sketches and then generate the view-dependent animations by moving the camera in the generated view-sphere. We are able to blend across multiple view-spheres to create complex motion sequences. We then compose animations generated by movement of multiple cameras to get two instances of a single character to perform as two different characters doing different actions in the same animation.*

## 1. Introduction

Stylized animation and rendering have been used, in recent years, to produce compelling and visually appealing imagery. Stylizations enhance the artistic impact of the animated character and impart them with a personality of their own. The stylization may vary the appearance of such characters with time, viewing direction, their mood or their pose. View-dependent geometry [16] is a form of stylization which allows the animator to make the character respond automatically to changes in the view direction. However, applying such stylizations to a character animation requires considerable skill and hard work on the part of the animator. We demonstrate a technique to reuse the view-dependent animation to create new stylized animations.

Synthesis of newer animations using motion synthesis techniques has been widely researched. All the previous work in the direction of animation reuse has generally focused on creating newer, meaningful motions given a database of previously recorded motion capture data. However, reuse of stylized animation is difficult because often the stylizations are generated for a particular viewpoint. View-dependent animation allows us to overcome this limitation.

View-dependent animations are responsive to changes in the camera. Hence, newer variations of the animation are generated automatically as the viewpoint changes. We present in this paper a way to synthesize new animations by aesthetically combining the variations generated due to changes in viewpoint. We start from sketches given by the animator, create view-dependent models using the system given by Chaudhuri et. al. [6], and generate the view-dependent animations by moving multiple cameras. This generates view-dependent variations of the animation. We give an example that illustrates one of the many ways in which the animator can combine/reuse the view-dependent variations to generate newer animation.

## 2. Background

Our work bridges across the two themes of stylized animation and animation reuse. We discuss the related work pertaining to these two areas one by one.

Several artwork styles have been explored in stylized

animation and rendering literature such as pen and ink, painting, informal sketching, and charcoal drawing. Many of these algorithms focus on a specific artistic style, or a closely related family of styles. For example, Meier [14] describes a technique for creating animations evocative of impressionistic painting, and Kowalski et. al. [12] create cartoon-style renderings reminiscent of Dr. Seuss. Hertzmann and Perlin [10] and Hays and Essa [9] give techniques to transform images and videos into painterly animations depicting different artistic styles. Stylizations based on innovative use of the camera have also been researched. Agrawala et. al. [1] present an interactive tool for creating multi-projection images and animation. Glassner [8] talks about using *cubist* principles in animation, i.e. rendering simultaneously from multiple points of view in an animation using an abstract camera model. Coleman and Singh [7] present an approach that distorts scene geometry so that when viewed through a standard linear perspective camera, the scene appears nonlinearly projected. They also address the impact of nonlinear projection on rendering and explore various illumination effects.

View specific distortions also form a very elegant method for specifying stylized animations. View-dependent geometry (VDG) as introduced by Rademacher [16] is a method for specifying models in such a manner that their *geometry can change with the view point*. The inputs to the system are a 3D model of a character (the base model) and a set of drawings of the character from various viewpoints. The user then manually matches the viewing direction and the shape of the base model with each drawing thus creating a *key viewpoint* and *key deformation* pair for every drawing. A key viewpoint and a key deformation pair together constitute a *view-dependent model*. All the key viewpoints taken together and projected onto a sphere around the base model form a *view-sphere*. Every key viewpoint has the corresponding key deformation associated with it. Tracing any camera path on this view-sphere generates the appropriate animation with view-dependent deformations. One of the major drawbacks is the amount of manual intervention required by the user in the creation of the view-dependent models. Chaudhuri et. al. [6] present a system that allows for automated recovery of the *key viewpoint* and creation of view-dependent models. We have augmented this system to create our animations, and we discuss the system in brief in the next section.

Synthesis of new animations from existing motion data has been extensively worked upon. Kovar et. al. [11] present Motion Graphs as a technique to encapsulate connections among the motion database. New motion can be generated by building walks on the graph. Lee et. al. [13] allow interactive control of avatars animated using motion data, with multiple interfaces to control the avatars behaviour. Arikan et. al. [2] allow the user to annotate a motion database and then paint a timeline with the annotations to synthesize newer motions. Rose et. al. [17] use radial basis functions to interpolate between and extrapolate around a set of aligned and labeled example motions (e.g., happy/sad and young/old walk cycles), then use kinematic solvers to smoothly string together these motions. Brand and Hertzmann [4] describe Style Machines to do stylistic motion synthesis by learning motion patterns from a highly varied set of motion capture sequences. Bregler et. al. [5] describe a method to capture the motion from a cartoon animation and retarget to newer characters.

**Contribution –** Our technique for synthesizing stylized animation is targeted more towards generating stylistic variations of the animations of a character depending on viewpoint changes rather than synthesizing new motion in general. The basis of view-dependent geometry provides us a setting to effect such variations very easily. We illustrate the concept through an example, but we would like to stress at this point that the example merely forms one of the *many* ways in which these stylistic variations can be reused. We will attempt to point out other ways to reuse the view-dependent stylizations while discussing one specific example in detail.

# 3. Overview

We use the system developed in [6] to create the view-dependent models. We describe the system in brief.

The system described in the original paper has two major components. We use the first component i.e. the *view-dependent deformation (VDD) system* to create the view-dependent models. The VDD system requires the following inputs: the sketches of the character in various poses and a 3D mesh model of the character which we call the *base mesh model*. The VDD system modifies the base mesh to match the sketched pose.

We embed a skeleton inside the mesh model using a simple interactive tool. We next construct a lattice made up of tetrahedral cells, enclosing the mesh, using another interactive tool. Then the system can recover a camera which best aligns the base mesh with the viewing direction in the sketch. For this the user needs to specify joint correspondences between the embedded skeleton and the sketched pose. Using these correspondences, the system can robustly solve for the best possible camera which matches the viewing direction in the 2D sketch of the character. The recovered camera is used to drive a view-dependent deformation model. The deformation model has a multi-layered structure. At the first stage it uses an inverse kinematics (IK) layer. This matches the pose of the embedded skeleton to the sketched pose. The IK layer is integrated with the view-dependent framework which constrains it to find poses very

close to the desired pose. At the second stage it uses a lattice based direct free form deformation (DFFD) layer. The mesh deformation model is also integrated with the recovered view, so the mesh can be deformed in accordance with the recovered view and a best possible match with the sketched character is obtained. This view recovery and deformation process is repeated for each of the sketched poses. At the end of this stage we have a set of recovered views and corresponding deformed meshes - these are the *key viewpoint* and *key deformation* pairs (introduced in the previous section) respectively.

The second major component of the system is the *view-dependent animation (VDA) system*. We have extended this component to help us create our animation. This set of *view-dependent models* is passed on to the VDA system. The animation system actually generates at run time the *view-sphere* and all the frames in response to the camera path traced on this sphere. We have extended the animation system to handle inter-view-sphere interpolations for the viewpoint and the generated view-dependent deformations, which allow us to expand the complexity of movements which the VDA system can handle.

In the following sections we present our techniques in detail. We first describe the generation of basic view-dependent stylization.

# 4. View-dependent stylization

After processing all the sketches using the VDD system, we have a set of recovered views and corresponding deformed meshes - these are the *key viewpoint* and *key deformation* pairs. To create view-dependent stylizations, we normalize all the recovered key viewpoints to a unit sphere which surround the base model. This is called the *view-sphere*. Then we partition the space inside the view sphere by computing its 3D convex hull (see Figure 2). We have used the Qhull [15] program which implements the quickhull [3] algorithm to compute the convex hull of the key viewpoints.

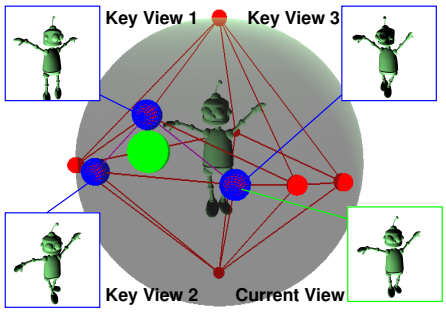



Figure 2: A view-sphere - The smaller blue and red spheres are the key viewpoints

## 4.1. Single view-sphere stylization

We want to get the deformed mesh model for any new viewpoint on this view-sphere. To get the desired deformation the system first finds the closest three *key viewpoints*. The new viewpoint will lie in the triangle formed by these three key viewpoints and hence will have barycentric coordinates, say given by $w_1$, $w_2$, $w_3$. To get the new deformed mesh at this new viewpoint, every vertex of the mesh (say $v$) is computed as a barycentric combination of the corresponding vertices in the three *key deformations* or deformed meshes associated with the afore mentioned key viewpoints (say $v^i$, $v^j$, $v^k$) as $v = w_1 \cdot v^i + w_2 \cdot v^j + w_3 \cdot v^k$. The barycentric interpolant is used here because it is linear and simple to compute. When one of the barycentric weights reaches a value of 1 and the other two weights are equal to zero, the current viewpoint exactly matches a key viewpoint (i.e., when the current viewpoint is at a triangle vertex). When this occurs, the new deformed model will exactly match the one key deformation corresponding to that viewpoint.

The barycentric weights can be scaled exponentially to alter the sharpness of the transition between adjacent deformations. We define these new, scaled weights as:

$$(w_i)^* = \frac{(w_i)^\alpha}{(w_1)^\alpha + (w_2)^\alpha + (w_3)^\alpha} \qquad (1)$$

If these $(w_i)^*$'s are used instead of the original $(w_i)$'s to generate the new deformed mesh for the current viewpoint, then as the parameter $\alpha$ grows $> 1$, the resulting blended model moves more quickly towards the nearest key deformation and as $\alpha$ becomes $< 1$, the blend is more gradual. Now animating in the view space consists merely tracing the camera path on the sphere and the animation is generated automatically in response to the view. An example of this can be seen in Figure 3, where the smaller green sphere represents the current viewpoint, while the smaller red spheres are the key viewpoints. The three closest key viewpoints are marked blue in every case. Note how the model deforms as the current viewpoint moves over the view-sphere (for clarity only the convex hull of the view space is shown instead of the view-sphere) The animation is specified in terms of a path on the view sphere and the animation system generates the view-sphere and the various meshes and novel view at run time.

## 4.2. Multiple view-sphere stylization

A single view-sphere cannot accommodate too many poses and hence the animations which can be generated using it are limited. For complex animation sequences the animator may generate multiple view-spheres (See Figure 4 which shows two different view-spheres). It should be noted that all the views of the second view-sphere cannot be put into the first view-sphere as that will cause the camera path

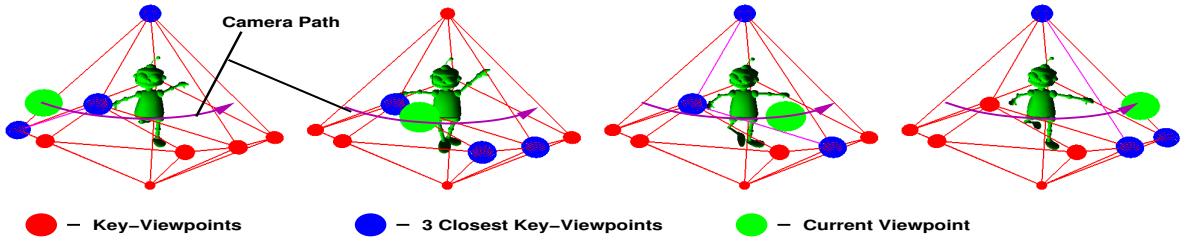



Figure 3: Moving the current viewpoint (green sphere) to generate the new deformed mesh

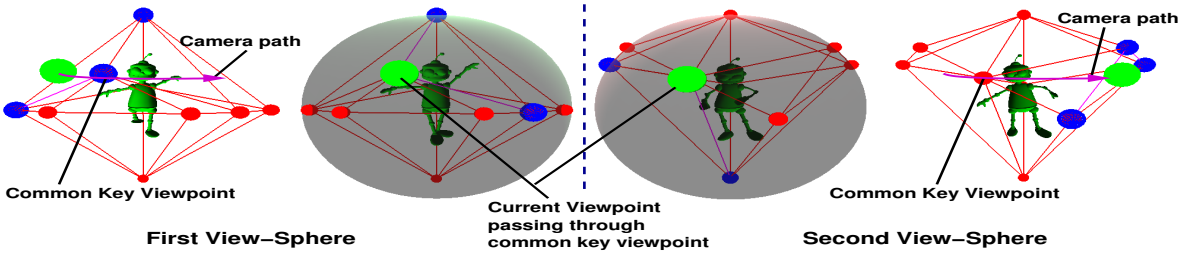



Figure 4: Interpolating among view-spheres with a common key viewpoint

planned on the first view-sphere to be unduly influenced (during interpolation for getting novel views) by the added views. This will make the generated animation different from what was conceived by the artist. Also, if we want to have different poses of the character from the same viewpoint, it is not possible to do this in a single view-sphere and requires us to work with multiple view-spheres. In cases where the base model is non-rigidly animated, a single set of key deformations is no longer sufficient. Such situations need a different set of key deformations for the keyframes of the model's animation. This essentially results in a separate view-sphere at each keyframe. The animation is generated by blending the deformations on a per-frame basis in response to the current viewpoint as the viewpoint moves from one view-sphere to another.

The challenge in working with multiple view-spheres lies in creating a seamless blend between the resulting animations as the viewpoint is transferred from one view-sphere to another. We explain how we have done a seamless blend for the example we have generated. Let $VS_1$ and $VS_2$ be the two view-spheres. Let $X = \{x_i : 0 < i \leq n\}$ and $Y = \{y_j : 0 < j \leq m\}$ represent the set of key viewpoints of the two view-spheres respectively. Now to interpolate seamlessly between $VS_1$ and $VS_2$, we require that $X \cap Y \neq \phi$ i.e. the two view-spheres share a common key viewpoint. Now if we animate along a path which passes through this key viewpoint it is merely a switching of view-spheres which effects a seamless blend between the animations generated by the two spheres. Note that only the key viewpoint needs to be same and not the key deformation. An example of this method is shown in Figure 4 where the view-sphere changes from the second to the third image, and the current viewpoint passes through the common key viewpoint between the two view-spheres (the current viewpoint is the small green sphere - note that it occupies the same position in space when the view-sphere changes). The animation generated by this movement of the viewpoint shown in the four images results in a smoothly blended animation. It should be noted that only swapping the view-spheres when the current viewpoint reaches the common key viewpoint will cause a discontinuity in the animation. If the common key viewpoint is $v_c$ and the key deformation corresponding to it is $d_1$ for $VS_1$ and $d_2$ for $VS_2$, then we swap $d_1$ with $d_2$ before the current viewpoint enters the triangle containing $v_c$. And once the current viewpoint is at $v_c$, then we swap $VS_1$ and $VS_2$.

This technique has the additional assumption that the base model has not been translated from $VS_1$ to $VS_2$. If any of the above assumptions do not hold then, we need to interpolate the mesh deformations between the point the current viewpoint leaves $VS_1$ and enters $VS_2$ using any standard interpolation technique like natural cubic splines.

# 5. A stylized animation example

The view-dependent stylizations can be put to myriad uses while making the animation. In the example we discuss, we create a ballet animation for a character named *Hugo*. We create an animation where we want two characters to perform a coordinated ballet sequence, however, both the characters are actually instances of the same character each performing different ballet moves.

## 5.1. Planning and Sketching

We begin by planning the storyboard for the animation and sketching the various key poses, for which we later create the view-dependent models, This stage is primarily governed by the artist's conception of the animation, the var-

ious poses and camera moves which will constitute the animation. The artist also needs to have an idea of how will the view-dependent stylization be used finally - here we use it to create the two characters. Some of the sketches we used are given in Figure 5. Taken pairwise the sketches will form the part of three different view-spheres. For example, Sketch 1 and 2 belong to the same view-sphere and they show how the character will look from two different viewpoints on that sphere. The animator also has to plan at this stage the camera movement direction during the animation and the point where the viewpoint will move from one view-sphere to another (see Section 4).

Figure 5: Sketches of various ballet poses

### 5.2. Reconstructing the Poses

Now the animator uses the VDD system to recover the cameras from the sketches which aligns the base mesh with the sketched pose, and then to deform the mesh to get the best possible match with the sketches (see Section 3). We show the posed meshes in Figure 6 for the sketches shown in Figure 5. The poses and the recovered cameras form the view-dependent models which are then passed on to our animation system.

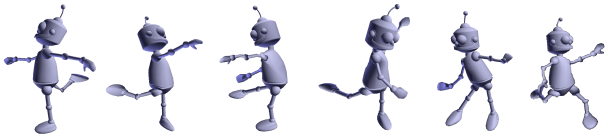


Figure 6: Hugo posed in various ballet poses

### 5.3. Rendering the animation

To get the desired animation, we plan the movement of two cameras across the view-spheres. Each of the cameras will generate one of our characters respectively (See Figure 7. Note that both the cameras cross over from one view-sphere to another in the centre view-sphere, while maintaining a smooth camera path). Further to enhance the illusion that we actually are animating two characters, we render the characters separately and compose the two camera views to get one coherent dance sequence. We show a sequence of frames from our final animation in Figure 8. The characters in green and red are our two dancers.

### 5.4. Other Possibilities

The idea for this animation was essentially inspired by *cubist* paintings which portray the parts of the same character in a painting from different perspectives, while we are showing two instances of the same character animation, each from different perspectives or viewpoints which lend them their unique view-dependent style. The animations are composed here to give a visually pleasing effect. The animations can be composed to form more abstract sequences as well, with different parts of the same character being given different view-dependent stylizations, controlled by their own cameras. Another application is to have one global camera control a crowd of characters and since the camera will be at a different location with respect to the local coordinate system of each character, each will be posed differently than the other. Another idea might be to control multiple cameras generating the view-dependent variations by the movement of one master camera and generate the animation as a weighted combination of the views of the controlled cameras, or from the viewpoint of the independent master camera.

We believe that the reuse of view-dependent stylizations can lead to many interesting animations which are not possible or are very cumbersome to create using other ways of doing animation.

## 6. Results

We have made the complete ballet animation. The animation changes four view-spheres during the dance sequence. Many characteristic ballet moves are performed by the character including an allégro, a pirouette and a promenade. The two characters perform different moves in tandem with each other, choreographed to be sometimes in response to each others moves and at other times be in competition with each other. We will like to stress here that all the moves for both the characters are always generated from the same view-sphere i.e. the two characters are always view-dependent variations of each other.

*The final animation titled "Through the Looking Glass" can be downloaded from our site at* `http://www.cse.iitd.ac.in/~parag/projects/style_reuse/`.

## 7. Conclusions

We have shown a technique for the stylized reuse of view-dependent animation to produce a unique, stylized animation sequence. We have shown how variation of poses of the same character generated in response to camera movements can be used to generate aesthetically pleasing animation.

It would be interesting to see other types of stylizations possible by reuse of view-dependent stylizations.

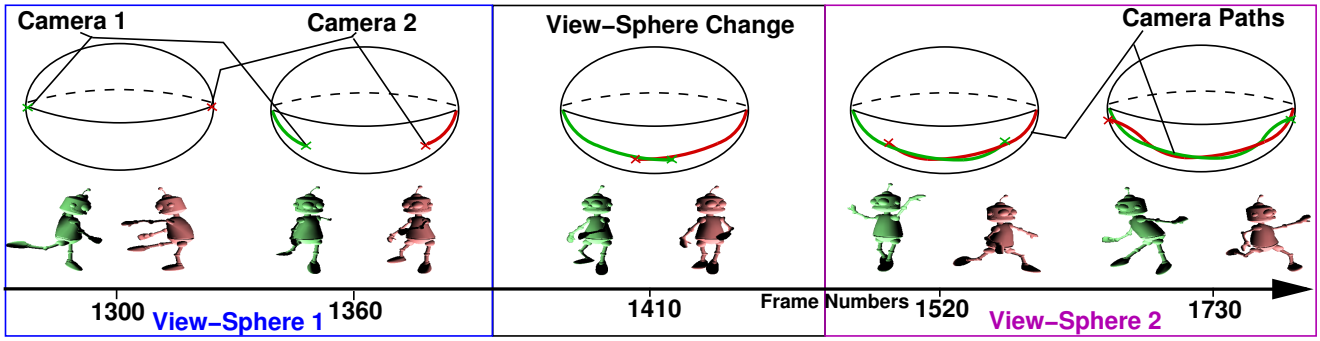



Figure 7: Camera 1 (Green) generates the green character, Camera 2 (Red) generates the red character. Each view sphere generates two character poses in response to the two cameras.

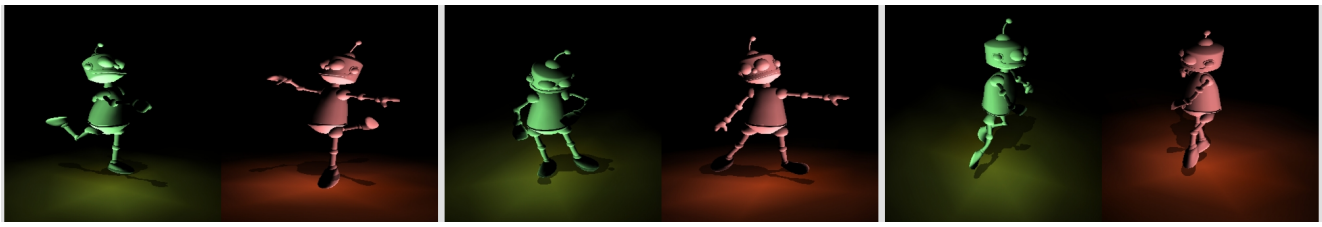


Figure 8: Frames from the completed animation

## Acknowledgements

Hugo's mesh is courtesy Laurence Boissieux, INRIA.

## References

[1] M. Agrawala, D. Zorin, and T. Munzner. Artistic multiprojection rendering. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 125–136, 2000.

[2] O. Arikan, D. A. Forsyth, and J. F. O'Brien. Motion synthesis from annotations. *ACM Transactions on Graphics*, 22(3):402–408, 2003.

[3] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

[4] M. Brand and A. Hertzmann. Style machines. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 183–192, 2000.

[5] C. Bregler, L. Loeb, E. Chuang, and H. Deshpande. Turning to the masters: Motion capturing cartoons. *ACM Transactions on Graphics*, 21(3):399–407, July 2002.

[6] P. Chaudhuri, P. Kalra, and S. Banerjee. A System for View-Dependent Animation. In *Eurographics*, 2004.

[7] P. Coleman and K. Singh. Ryan: Rendering your animation nonlinearly projected. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, pages 129–156, 2004.

[8] A. S. Glassner. Cubism and cameras: Free-form optics for computer graphics. *Technical Report (MSR-TR-2000-05)*, Jan. 2000.

[9] J. Hays and I. Essa. Image and video based painterly animation. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, pages 113–120, 2004.

[10] A. Hertzmann and K. Perlin. Painterly rendering for video and interaction. In *Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, pages 7–12, 2000.

[11] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 473–482, 2002.

[12] M. A. Kowalski, L. Markosian, J. D. Northrup, L. Bourdev, R. Barzel, L. S. Holden, and J. F. Hughes. Art-based rendering of fur, grass, and trees. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 433–438, Aug. 1999.

[13] J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard. Interactive control of avatars animated with human motion data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 491–500, 2002.

[14] B. J. Meier. Painterly rendering for animation. In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 477–484, Aug. 1996.

[15] Qhull. Convex hull computation software. http://www.thesa.com/software/qhull/.

[16] P. Rademacher. View-dependent geometry. In *Siggraph 1999, Computer Graphics Proceedings*, pages 439–446, 1999.

[17] C. Rose, M. F. Cohen, and B. Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–40, 1998.