The Cartoon Animation Filter

Jue Wang[1] Steven M. Drucker[2] Maneesh Agrawala[3] Michael F. Cohen[2]

[1]University of Washington [2] Microsoft Research [3] University of California, Berkeley

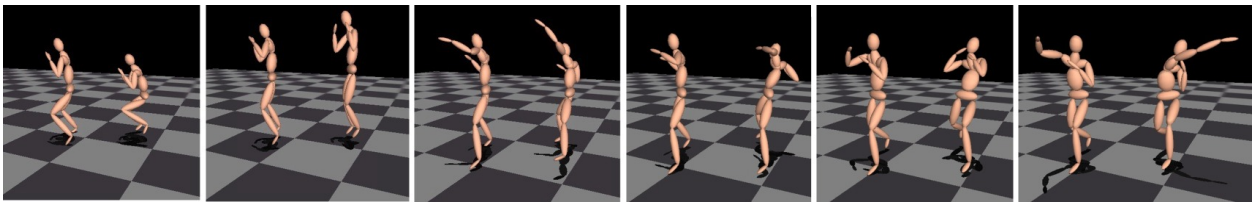

Figure 1: A punch. Left: before filtering. Right: after filtering.

Abstract

We present the "Cartoon Animation Filter", a simple filter that takes an arbitrary input motion signal and modulates it in such a way that the output motion is more "alive" or "animated". The filter adds a smoothed, inverted, and (sometimes) time shifted version of the second derivative (the acceleration) of the signal back into the original signal. Almost all parameters of the filter are automated. The user only needs to set the desired strength of the filter. The beauty of the animation filter lies in its simplicity and generality. We apply the filter to motions ranging from hand drawn trajectories, to simple animations within PowerPoint presentations, to motion captured DOF curves, to video segmentation results. Experimental results show that the filtered motion exhibits anticipation, follow-through, exaggeration and squash-and-stretch effects which are not present in the original input motion data.

1 Introduction

Cartoon animation is the art of manipulating motion to emphasize the primary actions while minimizing irrelevant movements. Expert animators carefully guide the viewers' perceptions of the motion in a scene and literally bring it to life. Some of the principles of cartoon animation are well known. As outlined in Thomas and Johnston's "The Illusion of Life" [1995] and repeated by Lasseter [1987], skilled animators add features not seen in the real world, including anticipation and follow-through to the motion with related squash and stretch to the geometry. Yet, most of us do not possess the time and skills necessary to keyframe such effective animated motion by hand. Instead, we create very basic 2D motion paths in tools like PowerPoint or we rely on recording devices such as video cameras or motion capture (MoCap) systems to faithfully record *realistic* motion. However, such motions often appear disappointingly wooden and dead in comparison to good cartoon animation.

In this paper we present a simple yet general *cartoon animation filter* that can add both anticipation/follow-through, and squash and stretch, to a wide variety of motion signals. Mathematically, the filter can be described as:

$$x^*(t) = x(t) - \overline{x''}(t) \quad (1)$$

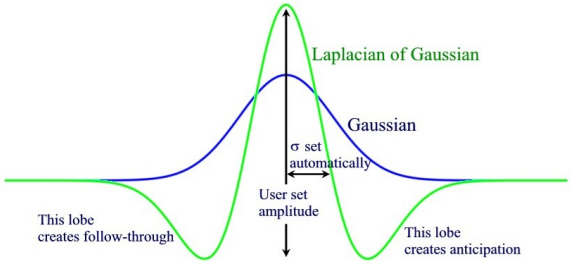

Figure 2: The cartoon animation filter.

where $x(t)$ is the signal to be filtered, and $\overline{x''}(t)$ is a smoothed (and possibly time shifted) version of the second derivative with respect to time of $x(t)$. This approach is equivalent to convolving the motion signal with an inverted Laplacian of a Gaussian (LoG) filter.

$$x^*(t) = x(t) \otimes -LoG \quad (2)$$

The filter provides both simplicity and speed. While the filter cannot always produce the exact motion a skilled animator would create, it provides a tool that can be coded quickly and then applied to a broad class of existing motion signals. It is fast enough for online applications such as games.

Although the idea of enhancing animation by filtering the motion signals [Unuma et al. 1995; Bruderlin and Williams 1995] is not in itself new, our work extends these previous techniques and makes three novel contributions:

Unified approach. We use the same filter to produce anticipation, follow-through and squash and stretch. Anticipation and follow-through are a natural byproduct of the negative lobes in the LoG filter. Applying the filter with spatially varying time shifts across objects generates squash and stretch.

One parameter interface. We automatically generate good default values for almost all of the parameters of our filter, leaving the user to specify only the overall strength of the exaggeration. Our approach allows novice users to easily enhance and enliven existing motions.

Variety of motion signals. We demonstrate that our filtering approach can enhance several types of motion signals including video recordings, MoCap, and simple path-based motions created with PowerPoint.

The cartoon animation filter is simple enough to operate on motion curves in real time, thus motions designed for games, such as

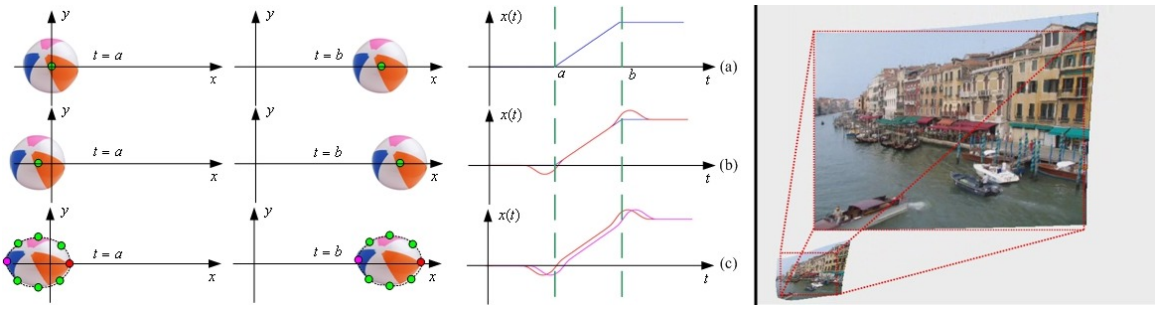

Figure 3: Left: (a). A simple 1D translational motion on a ball. (b). Filtered motion on the centroid of the ball, the filter adds anticipation and follow-through effects to the motion. (c). By applying the filter on the outline vertices with time shifts, the ball exhibits the squash and stretch effects. Right: Two superimposed frames of a PowerPoint animation after the animation filter is applied. The dotted lines show the original path.

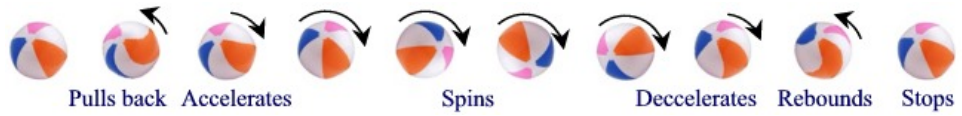

Figure 4: A ball starts at rest, then spins, then stops. The filter creates an anticipatory pull back before the spin and a follow-through rebound.

the boxer in Figure 1, could be modified programmatically by dynamically adjusting the filter strength to convey the apparent energy level of the character.

There are many interactive animation design systems to construct an initial animation ranging from professional tools to informal systems designed for quickly creating animation, such as the Motion Doodles system [Thorne et al. 2004]. Our work would serve as a complement to such systems.

2 Related Work

While the importance of cartoon style exaggeration is well recognized, none of the previous techniques combine a single unified approach with a simple one parameter user interface. For example, simulation, both physically-based [Faloutsos et al. 1997] and stylized [Chenney et al. 2002; Igarashi et al. 2005], as well as surface deformation [Wyvill 1997] are common techniques for generating squash and stretch. These techniques do not handle anticipation and follow-through. They also force users to set many parameters and are complicated to implement. Moreover these techniques require an underlying 2D or 3D model and therefore cannot be directly applied to some types of motion signals including video.

Motion signal processing is another approach for exaggerating motion that applies signal processing techniques to motion curves. Our methods lie in this class of techniques. Unuma et al. [1995] introduced the approach and used it to interpolate and extrapolate between MoCap walking sequences (e.g. brisk, tired, fast, ...) in the frequency domain. Bruderlin and Williams' [1995] also process MoCap data in the frequency domain. Adopting the user interface of a graphic equalizer they provide slider controls over gains of frequency bands for joint angles. While they demonstrate that carefully adjusting particular frequency bands can generate anticipation and follow-through, the controls are extremely low level and unintuitive. Users must mentally map controls over frequency bands into effects in the time domain. They also limit their techniques to MoCap data only and do not address squash and stretch effects.

Collomosse's VideoPaintbox [2004] includes a combination of techniques for genereating cartoon style animation from video. After segmenting the video into individual objects, a complex case-based algorithm with several (4 to 6) user set parameters is used to modify the motion curves and generate anticipation and follow-through. A completely separate deformation-based approach is used to squash and stretch the objects. While the VideoPaintbox can generate nicely stylized video motions, the complexity of both the implementation and interface make it difficult to enhance motion quickly. Campbell et al [Campbell et al. 2000] show examples of adding flexibility to animated objects.

In an inspiration for this work, Liu et al. [2005] demonstrate a motion magnification system to extract small motions from video and multiply the magnitudes of the motion by a constant for re-rendering an exaggerated video. We show that applying our filters on video objects create similar exaggeration with additional effects such as anticipation and follow-through.

3 Motion Filtering

3.1 The Filter

The cartoon animation filter, as defined in the introduction, subtracts a smoothed (and potentially time shifted) version of the motion signal's second derivative from the original motion. More specifically, the second derivative of the motion is convolved with a Gaussian and then subtracted from the original motion signal.

$$\overline{x''}(t) = x''(t) \otimes Ae^{(-(((t/\sigma)\pm\Delta t)^2))} \quad (3)$$

where $x''(t)$ is the second derivative of $x(t)$ with respect to time, and the amplitude, A, controls the strength of the filter. A second parameter, σ controls the Gaussian standard deviation, or width, of the smoothing kernel. As we will show, the time shift, Δt, can be used to create stretch and squash effects by applying the filter shifted forward in time for the leading edge of the acceleration and shifted backward in time for the trailing edge.

Equation 2 provides a more compact and efficient filter based on the (Laplacian) of the Gaussian, LoG. We implement the cartoon animation filter as such. The inverted LoG filter, is similar to the *unsharp* filter[1]. in the image domain. As with any sharpening filter, it produces a *ringing* which is often considered an undesirable artifact in images. In our case, the same ringing produces a desired anticipation and follow-through effect.

[1]The unsharp filter is often defined as the difference of Gaussians, or DoG filter, which is similar is shape and effect to the LoG.

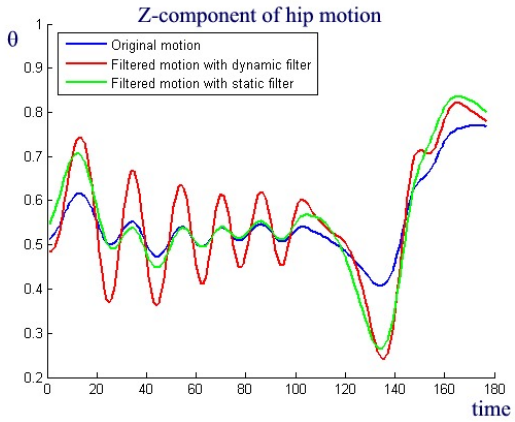

Figure 5: By dynamically adapting σ the animation filter is able to exaggerate all parts of the motion.

Figure 3 shows an example of applying the filter to a simple translational motion. In this example a ball stands still, then moves with constant speed to the right, then stops. By applying the cartoon animation filter with no time shift, to the centroid of the ball we add anticipation and follow-through to the motion (i.e., it will move slightly to the left of the starting location before going right, and will overshoot the stop point). These effects are due to the negative lobes on the inverted LoG filter that precede and follow the main positive lobe in the center of the filter (see Figure 2).

In principle an expert user could manually set any of the parameters (A, σ, Δt) of the filter, we have developed automated algorithms for setting σ and Δt so that novice users can simply specify the strength of the filter A.

3.2 Choosing the Filter Width

The width of the filter is defined by the standard deviation, σ, of the Gaussian. Intuitively, we would like the dominant visual frequency, ω^*, of the motion to guide the frequency response of the filter. As the frequency changes, we would like the filter width to change as well. To do this we use a moving window over 32 frames centered at t and take the Fourier transform of the motion in the window. We then multiply the frequency spectrum by the frequency and select the maximum as the dominant visual frequency, ω^*.

$$\omega^*(t) = \max_\omega \ |X(\omega)|\omega \tag{4}$$

Or equivalently, we can take the maximum of the Fourier transform of the velocity $x'(t)$.

$$\omega^*(t) = \max_\omega \ |\mathcal{F}(x'(t))| \tag{5}$$

Equation 5 expresses the fact that we are concerned with the dominant frequency in the velocity changes. The width of the LoG filter thus varies over time and is defined by $\sigma(t) = 2\pi/\omega^*(t)$.

Figure 5 illustrates how the dynamically modified σ is able to exaggerate all parts of the motion in a uniform way. The blue curve shows the Z-component of hip motion of a "golfswing" MoCap sequence. As we can see the dominant frequency of the motion dynamically changes overtime. A fixed width LoG filter (the green curve) exaggerates the latter part of the motion but fails to consistently modify the earlier portion. By dynamically setting σ our filter exaggerates the motion throughout the animation.

3.3 Squash and Stretch

Deformation is another important effect in animation to emphasize or exaggerate the motion. Squash and stretch is achieved by slightly time shifting the same LoG differentially for the boundary points of the object.

We use the ball shown in Figure 3 to illustrate the idea, and the same approach can be applied to more complicated shapes. Instead of representing the object using its centroid, we represent it by a set of vertices along its outline, as shown in Figure 3c. For a vertex p, we calculate the dot product of a vector from the centroid to the vertex, $\vec{B_p}$, normalized by the maximum length vector, and the acceleration direction as $s_p = \tilde{\vec{B}_p} \cdot |\vec{x''}|$, and time shift the LoG filter based on s_p as

$$LoG_p(t) = LoG(t - \Delta t) \tag{6}$$

where

$$\Delta t = s_p \cdot \sigma(t) \tag{7}$$

$$\tilde{\vec{B}_p} = \vec{B_p}/max_p B \tag{8}$$

When $s_p > 0$ (the red vertex in 3c), the vertex is on the leading edge of the acceleration will anticipate the upcoming acceleration. On the contrary, if $s_p < 0$ (the purple vertex in 3c), the vertex is on the trailing edge of the acceleration thus it will be effected later. Since we add time shifts differentially to each vertex, the object will deform, as shown in Figure 3c. Rotational acceleration is treated similarly by tessellating the shape and applying the time shift to both internal and external vertices resulting in a twisting anticipation and follow-through (see Figure 4).

3.4 Area Preservation

At each point in time, the difference in the deformation between the leading and trailing vertices will create a stretch or squash effect along the line of acceleration. To approximately preserve area we scale the object in the direction perpendicular to the acceleration inversely to the squash or stretch.

4 Results

The cartoon animation filter is independent of the underlying representation of the motion, and can be applied to a variety of motion signals. We demonstrate the filter on three types of motions: hand drawn motion trajectories, segmented video objects and MoCap DOF curves of linked figures.

4.1 Filtering Hand Drawn Motion

Basic animation can be created by hand drawing a 2D motion curve and having a 2D object follow the path. For example, PowerPoint allows users to attach motion curves (hand drawn or preset) to an object. The same filter can be applied to these objects as shown on the right side of Figure 3.

The cartoon animation filter provides an easy method to enliven such a simple motion. As shown in Figure 3, our filter can simultaneously add anticipation, follow-through and deformation effects to a simple translational motion. The centroid, and each vertex defining the moving shape are filtered independently. We describe in the next section how to carry the object texture along the modified motion.

4.2 Filtering Video Objects

To apply the filter to the motion presented in a video sequence, we first extract video objects using the interactive video cutout system

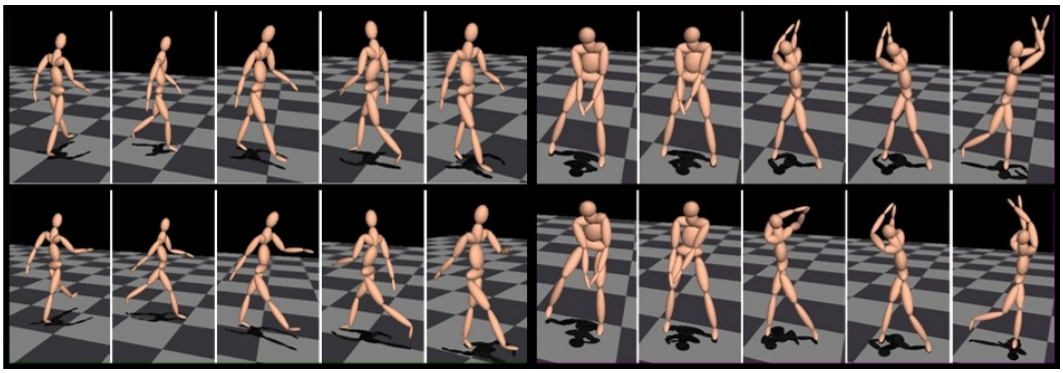

Figure 6: Applying the filter to two MoCap sequences: walk(left), and golfswing(right). Top: original motion. Bottom: filtered motion.

[Wang et al. 2005]. Once we segment out the object region on each frame t, we parameterize the outline into a polygon S_t and use the set of polygons as the representation of the motion, as shown in Figure 8. We then apply the animation filter to the centroid of the polygons, and the time shifted filter to each vertex based on acceleration of the centroid and the vector from the centroid to the vertex.

Maintaining constraints. This simple application of the animation equation will often result in deviations from sematic constraints. For example, the skateboarder may no longer be standing on the skateboard. To maintain constraints, we specify that the vertices on the bottom of the feet must retain their original paths. For each constrained vertex, the difference between the constrained position and the position after filtering is spread to the other vertices with weights inversely proportional to the distance from the constrained vertex.

Texturing deformed objects. To texture a deformed video object, we first triangulate the original outline to create a 2D object mesh [Shewchuk 2002]. For each vertex inside the object, we compute a mean value coordinate [Floater 2003] based on its relative location to the vertices on the outline. Once the outline is deformed, we use the calculated mean value coordinates to re-locate each vertex inside the object, resulting in a deformed object mesh (Figure 8d). We then perform linear texture mapping to create a deformed object based on the deformed mesh.

The filter stretches the skateboarder when he jumps onto the chair, and squashes him when he lands on the ground. These squash and stretch effects significantly exaggerate the motion and make it more alive. Figure 7 shows another example of filtering a video object in which the girl stretches on the downswing and compresses on the upswing.

4.3 Filtering MoCap Data

The same animation filter works well when applied independently to the individual degree-of-freedom(DOF) motion curves from a MoCap session. The human skeleton model we use has 43 degrees of freedom and we apply our filter independently to each DOF except the translation of the root node in the directions parallel to the ground plane. Figures 1 and 6 show the results of applying the filter to three different MoCap sequences. In all the examples we set the amplitude of the filter to be 3 and σ is dynamically adjusted for each DOF throughout the sequence. The filter is fast enough to be applied in real-time, thus the amplitude of the filter can be modified in online settings such as games to reflect the character's momentary energy level.

Maintaining constraints. Motion captured data implicitly exhibits constraints such as the fact that feet do not (generally) slide when in contact with the floor. Much like the constraint maintenance for video objects we add inverse kinematic constraints when applying the cartoon animation filter to MoCap data. We have only implemented the simplest of inverse kinematics that translates the root node to maintain constraints. For example, at foot down we record the foot position and assure the foot stays put by adjusting the root translation at each frame. Ideally, one should use more modern inverse kinematic methods after filtering to generate a smoother result.

Figure 7: Applying the filter to the monkeybar sequence. Top: original frames. Bottom: corresponding frames with filtered motion.

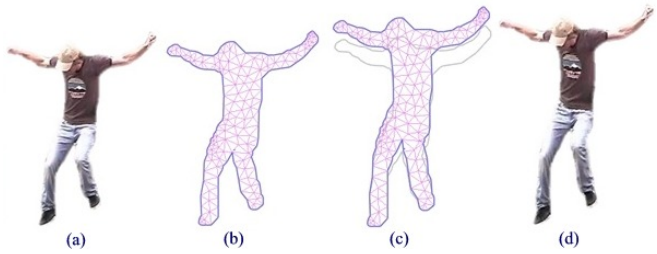

Figure 8: Illustration of deforming video objects. (a) Extracted object. (b) Parameterized object mesh. (c) Deformed object mesh. (d) Deformed object by texture mapping.

Squash and stretch. For the motion capture sequences, we do not have only a single shape to deform. For simplicity, we choose to only examine the vertical motion of the root to create squash and stretch. Filtering the root's vertical motion places the root at times higher and lower than in the unfiltered motion. We simply scale the whole figure vertically based on the ratio of the filtered height from the floor vs. the unfiltered height.

5 Conclusion

We have demonstrated a simple, one parameter filter that can simultaneously add exaggeration, anticipation, follow-through, and squash and stretch to a wide variety of motions. We have tried to maintain a balance between simplicity and control that favored simplicity. Thus, the application of the cartoon animation filter probably is not satisfactory for hand crafted off-line animation systems although it may be useful for previews. We believe the value of such a simple approach will be in either realtime applications such as games, or in less professional settings such as a child focused animation tool, or in 2D presentation systems such as PowerPoint.

Acknowledgements

The authors would like to thank Bill Freeman, Brian Curless, and Simon Winder for valuable discussions, and Keith Grochow for his help on the MoCap data and system. The first author was supported by Microsoft Research.

References

BRUDERLIN, A., AND WILLIAMS, L. 1995. Motion signal processing. In *Proceedings of SIGGRAPH 95*, 97–104.

CAMPBELL, N., DALTON, C., AND MULLER, H. 2000. 4d swathing to automatically inject character into animations. In *Proceedings of SIGGRAPH Application Sketches 2000*, 174–174.

CHENNEY, S., PINGEL, M., IVERSON, R., AND SZYMANSKI, M. 2002. Simulating cartoon style animation. In *NPAR 2002: Second International Symposium on Non-Photorealistic Rendering*, 133–138.

COLLOMOSSE, J. 2004. *Higher Level Techniques for the Artistic Rendering of Images and Video*. PhD thesis, University of Bath.

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 1997. Dynamic free-form deformations for animation synthesis. *IEEE Transactions on Visualization and Computer Graphics 3*, 3 (July - September), 201–214.

FLOATER, M. S. 2003. Mean value coordinates. *Computer Aided Geometric Design 20*, 1, 19–27.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics 24*, 3, 1134–1141.

JOHNSTON, O., AND THOMAS, F. 1995. *The Illusion of Life: Disney Animation*. Disney Editions.

LASSETER, J. 1987. Principles of traditional animation applied to 3d computer animation. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, 35–44.

LIU, C., TORRALBA, A., FREEMAN, W. T., DURAND, F., AND ADELSON, E. H. 2005. Motion magnification. In *Proceedings of SIGGRAPH 2005*, 519–526.

SHEWCHUK, J. R. 2002. Delaunay refinement algorithms for triangular mesh generation, computational geometry: Theory and applications. *Computational Geometry: Theory and Applications 22*, 1-3, 21–74.

THORNE, M., BURKE, D., AND VAN DE PANNE, M. 2004. Motion doodles: an interface for sketching character motion. *ACM Transactions on Graphics 23*, 3 (Aug.), 424–431.

UNUMA, M., ANJYO, K., AND TAKEUCHI, R. 1995. Fourier principles for emotion-based human figure animation. In *Proceedings of SIGGRAPH 95*, 91–96.

WANG, J., BHAT, P., COLBURN, A. R., AGRAWALA, M., AND COHEN, M. F. 2005. Interactive video cutout. In *Proceedings of SIGGRAPH 2005*, 585–594.

WYVILL, B. 1997. *Animation and Special Effects*. Morgan Kaufmann, ch. 8, 242–269.