CAMERA MOTION STYLE TRANSFER

Christian Kurz[1] **Tobias Ritschel**[2] **Elmar Eisemann**[2] **Thorsten Thormählen**[1] **Hans-Peter Seidel**[1]

[1]MPI Informatik, Saarbrücken, Germany
[2]Télécom ParisTech / CNRS-LTCI, Paris, France and Intel Visual Computing Lab, Saarbrücken, Germany

Abstract

When depicting both virtual and physical worlds, the viewer's impression of presence in these worlds is strongly linked to camera motion. Plausible and artist-controlled camera movement can substantially increase scene immersion. While physical camera motion exhibits subtle details of position, rotation, and acceleration, these details are often missing for virtual camera motion. In this work, we analyze camera movement using signal theory. Our system allows us to stylize a smooth user-defined virtual base camera motion by enriching it with plausible details. A key component of our system is a database of videos filmed by physical cameras. These videos are analyzed with a camera-motion estimation algorithm (structure-from-motion) and labeled manually with a specific style. By considering spectral properties of location, orientation and acceleration, our solution learns camera motion details. Consequently, an arbitrary virtual base motion, defined in any conventional animation package, can be automatically modified according to a user-selected style. As shown in our experiments, the resulting shots are still fully artist-controlled, but appear richer and more physically plausible.

Keywords: style transfer, camera shake, camera motion estimation, structure-from-motion

1 Introduction

The ability to change the camera viewpoint over time – *camera motion* – is a key ingredient of expressive film making. Different movies, TV productions, or cinematographers often adhere to specific styles of camera motion. Some famous directors even have a stylistic "trademark" on how they employ camera motion [1].

The style in which a camera is moved through the scene can substantially change the perception of the scene. Different camera motion styles are often used to highlight certain moods or situations and can be an important narrative element, e.g., fast-paced camera motion in action scenes. In some movies, e.g., *Cloverfield* or *Blair Witch Project*, a shaky camera style adds to the impression of the story being non-fictional and the scenes being recorded by the protagonists themselves with a consumer hand-held camera.

In most real-world productions, however, strong camera shake is not desired and a multitude of different equipment is used to reduce the high frequencies in the camera motion paths. This is achieved by steadicam systems or by mounting the camera to a dolly, a boom, or a crane. But even well-stabilized camera systems still exhibit shake. Observers got used to these more or less subtle deviations, and tend to classify scenes without camera shake as unnatural.

Camera animations in virtual worlds are usually generated by an animator who designs a camera motion by assembling a motion path from multiple mathematically-defined smooth motion curves. These motion curves contain no camera shake at all. In a virtual scene the animator has more artistic freedom in designing the camera path and can test more options than a cinematographer at a real-world set. Nevertheless, the results are often not convincing and have an artificial appearance.

Creating a convincing camera movement manually is a challenging task. Therefore, in this paper we present an approach to transfer camera shake from real videos onto an artificially designed camera motion path in a virtual scene. These additional subtle or strong movements increase the feeling of unpolished, direct realism. The resulting virtual camera motions can still be designed by an artist, but exhibit a richer and more physically plausible look.

Our solution consists of three stages: the data acquisition stage, the learning stage and the query stage. In the initial data acquisition stage, our system extracts camera motion from a video database using computer vision techniques. In the second

stage, the learning stage, we decompose this camera motion into a base and a detail component and store it, so that base motions can define a query for suitable detail motions. In the final query stage, a user provides base motions using common animation techniques, like three-dimensional spline editing, and the system adds suitable details interactively. A virtual path with plausible details can thereafter be used to drive a camera in a virtual scene.

The paper is structured as follows. We review related work in Section 2. Section 3 gives details about our approach. We present results in Section 4 that are discussed in Section 5 before concluding in Section 6.

2 PreviousWork

In this paper, we combine ideas from camera control and animation processing, with a recent trend in rendering and image processing which seeks to extract visual style.

Our system relates to camera control for static [2, 3, 4] or animated scenes [5, 6, 7], and overlaps with storytelling [8] in a continuous range from complete artist camera control to complete computer control. Our approach is orthogonal to and could be combined with such approaches because we maintain the original artist-defined path and only enrich it by transferring camera motion styles.

The formation of images using a certain camera is described by its internal parameters. Projective parameters can be determined using images of known scenes [9] and image sequences allow to capture the radiometric response [10]. Recently, image collections [11] were used to calibrate brands of cameras. A separation of base and detail information of camera motion of real observations has not been described yet.

Camera shake can be an unwanted effect and can be removed using camera stabilization [12]. Buehler et al. [13] extracted 3D camera motion using structure-from-motion to de-shake and then warp the images according to this new motion. Content preserving warps can be used to reliably warp the image into the stabilized 3D view [14]. After extracting a 3D camera motion path, Gleicher and Feng [15] project this path onto the closest "cinematographic" path to make a video appear more directed.

Moving a camera during its (virtual) film exposure also influences the scene capture and leads to motion blur [16]. For camera shake, the movement is complex, leading to a complicated blur kernel and, hence, a distinct look (cf. Fig. 1)

Separating a signal into *style* (sometimes also called the "form", the "look" or the "decoration") and *content* (sometimes called "base" or "guide") is of interest in many classic media, such as traditional painting [17], or even dancing, where variations are added on top of a basic movement. Tackling this challenge computationally allows, among other applications, to perform *style transfer*, that is, to extract a style from a signal and apply it to different content.

Image analogies [18] learn the style of a certain image filter by analyzing several before-after examples, similar to approaches in texture synthesis. Adjustments of tonal values (histogram matching) [19, 20] are often successful in reproducing a distinct photographic look.

Figure 1: Four frames from an animation with complex blur kernels generated by our system for a walking style motion.

Surfaces, like landscapes, also have distinct statistical properties that follow a guide signal [21]. For the special case of human face geometry Golovinskiy et al. [22] derived a statistical model to synthesize details.

Style transfer has also been applied to animation, e. g. , of humans [23, 24, 25], to transfer motion between characters [26] or to cartoon characters [27].

Texture synthesis can be seen as extracting only the decoration and then re-synthesizing it to fill a certain space [28, 29, 30, 31]. In [32], Mertens et al. synthesize texture details, but use the underlying geometry as a guiding signal.

3 Camera Motion Style Transfer

Our system for camera motion style transfer consists of three phases: the data-acquisition phase, the learning phase and the query phase. During the acquisition phase, a database of camera motions is built from a selection of input videos containing different types of camera motion. Thereby, the camera motion information is extracted from the videos by a fully-automatic camera motion estimation algorithm. In the learning phase, the camera motion paths are separated into low and high frequencies (base and detail layer). In the query phase an animator can specify a query-camera motion path using a standard modeling tool, e.g., by key-framing or, in our case, by editing three-dimensional splines. The base layer of the query camera motion

is compared to the base layers of the videos in the database. The detail layers of the locally-well-matching paths are used to add suitable details to the stylized camera path (cp. Fig. 2).

3.1 Acquisition phase

For the data acquisition we used selected videos of varying camera motions from different sources, including casual home video, feature films and TV series. All input videos had a sample rate of 25 Hz. We exclude all forms of virtual camera motions, such as computer animated footage.

From this data, we reconstruct the three-dimensional camera-motion paths originally used in the shooting of the footage. Our fully automatic camera motion estimation approach [33] employs well-known computer vision techniques [34, 35, 36] to extract the camera position, orientation, and focal length for each frame of the video. As the absolute scale of the scene cannot be determined from the videos, we convert the automatically obtained result into an absolute framework by manually specifying one spatial reference unit for each video. The resulting camera motion is a regularly-sampled time series of camera positions and orientations. Such a representation has six degrees of freedom per frame, where position is stored as an Euclidean 3-vector $\mathbf{x}$ and orientation as a unit quaternion $\mathbf{q}$. We denote the i-th out of N camera motion paths representing positions and orientations as a time-varying mapping from time t to position $\mathbb{R}^3$ and orientation $\mathbb{S}^3$:

$$f_i(t) : t \rightarrow \mathbb{R}^3 \times \mathbb{S}^3 \quad .$$

In practice, f is only available over a limited time range $[0, t_{max}]$. If required, we extend the definition of f to $\mathbb{R}$, by specifying an *border mode*, such as *repeat* (the default) or *ping-pong*. Further, we denote the matrix corresponding to a transformation $(\mathbf{x}, \mathbf{q})$ as $T(\mathbf{x}, \mathbf{q})$ and the inverse as T^{-1}. The resulting data is called the *camera-motion database*. While the algorithms also extract the focal length, we do not make use of this variation and keep all intrinsic parameters unchanged during the stylization.

3.2 Learning phase

In the learning phase, we use the camera-motion database to establish a relation between *base* camera motion f^{base} and camera-motion *details* denoted as f^{detail}, which we call the *camera-style model*. This model predicts the detail statistics of a certain base motion with respect to a chosen camera style at a certain point in time. Precisely, the base motion will act as the key and the details as the value of a style relation. We decompose the camera motion into base and detail as follows (cf. Fig. 3).

Base The base motion, is a smooth version of the camera path, i. e., the basic curving of the camera path. For smoothing, we use a Taubin filter [37] $\mathcal{S}$ of approximately 0.5 seconds support:

$$f_i^{\text{base}}(t) = \mathcal{S}(f_i)(t) \quad .$$

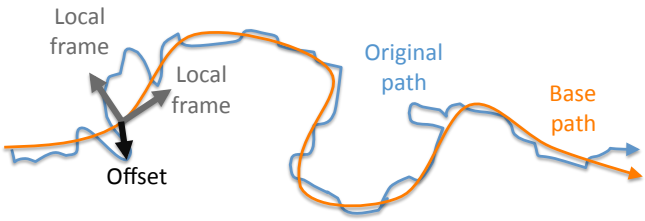

Figure 3: We decompose the original camera animation path into a smooth base path as well as details which are stored in the local frames of the base path.

In other words, all motion finer than 0.5 seconds is considered a detail, all coarser motions are included in the base motion.

Taubin's filter is a non-shrinking filter, which is important for the two boundary vertices at 0 and $t_{\max}$. We use 100 iterations with simple (1,2,1) weights.

Detail Next, we define the detail camera motion as the original input camera pose, but in the reference frame of the base path (an offset transformation)

$$T(f_i^{\text{detail}}(t)) = T^{-1}(f_i^{\text{base}}(t)) \cdot T(f_i(t)) \quad .$$

E. g., a camera motion that is already smooth will not lead to any details at all, whereas areas with a strong deviation between the smooth and the non-smooth version of the path have more details – as expected. Note, that the non-shrinking property of the Taubin filter [37] prevents spurious detail values at the boundaries. Further, shrinkage would influence the extraction and shift base content in the detail layer.

Finally, we perform local frequency analysis (Gabor decomposition) of this offset transformation leading to a time-varying spectrum:

$$F_i^{\text{detail}}(\omega)(s) = \mathcal{F}(f_i^{\text{detail}}(t) \cdot N_\sigma(t - s))$$

A Gabor transform is a Fourier transform $\mathcal{F}$ with a time-varying parameter s. This parameter is used to mask the input signal's intensity with a Gaussian and give more weight to the part of the signal that is close to s in time.

Fig. 4 shows a Gabor decomposition for a very simple input signal. For Gabor filtering, we use a σ of around 0.2 units, that is, at every point in time we locally consider the frequencies of 0.1 units to the left and 0.1 units to the right in time.

Query Key As a key to query details, we use the derivative of the base:

$$f_i^{\text{key}}(t) = \nabla f_i^{\text{base}}(t) \quad .$$

We compute the derivative ∇ using simple forward differentiation, which works well as $f_i^{\text{base}}(t)$ is already smooth. We use the derivative because it correlates best with the detail movements. For example, moving faster (large

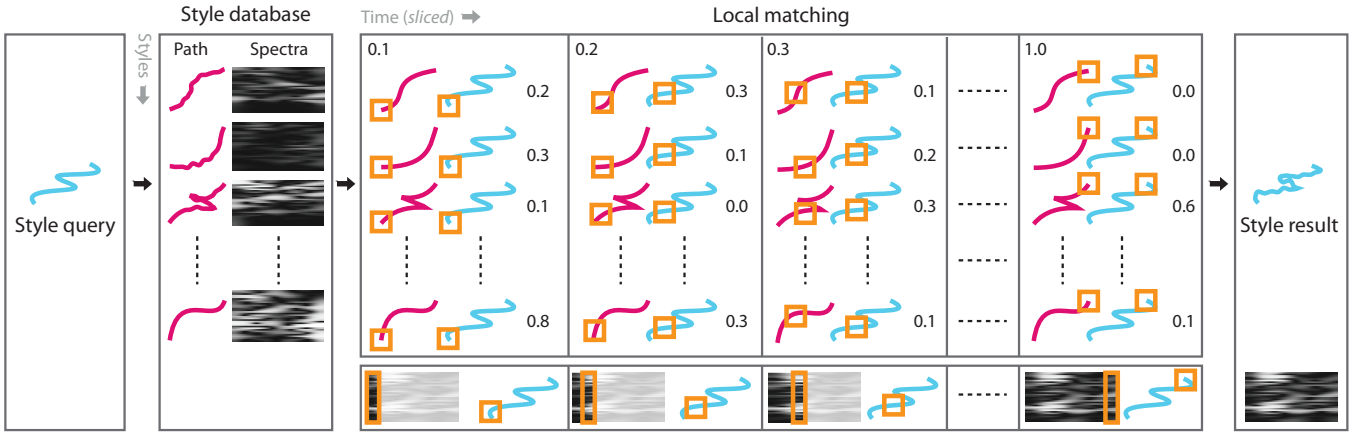

Figure 2: Stylization overview: Input to the stylization is a smooth virtual camera path (blue line): the style query. A style database of camera paths (red lines) and their time varying spectra (blocks) was build in a pre-process. To perform the query, the camera path (blue line) is locally matched against smooth versions of all training examples (pairs of red and blue paths) at every point in time (orange square) resulting in a distance (number). For each time slice, a new spectrum (orange block at bottom) is generated as a linear combination of the database spectra, weighted by the inverse distance. Note, that orange blocks are slices in time: When fixing time, time-varying spectra are just spectra. Finally, we produce the style result by adding noise with a matching time-varying spectrum.

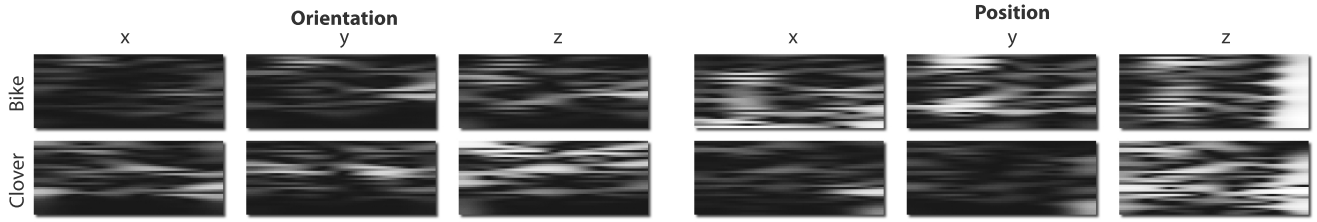

Figure 5: Example spectra extracted from two camera motion paths using a Gabor transform. The top row shows the spectra of a path obtained from a sequence filmed while making a right turn with a bicycle; the bottom row depicts the spectra of a path reconstructed from a running sequence in the movie **Cloverfield***. From left to right: position, x-, y-, and z-component; orientation, x-, y-, and z-component.*

derivative) gives stronger details, as well as moving around a corner (derivative with multiple non-zero components) results in more shake than a straight movement (only a single non-zero component).

3.3 Query Phase

In the query phase, a user provides a base camera movement $g^{\text{base}}(t)$. Its derivative $g^{\text{key}}(t)$ is matched against all elements f_j^{key} in the database of the chosen style. Each database element has an associated spectrum $F_i^{\text{detail}}(\omega)(s)$ that encodes the detail statistics. These statics are linearly weighted to yield the detail statistics $G^{\text{detail}}(\omega)(s)$ that are synthesized by:

$$G^{\text{detail}}(\omega)(s) = \sum_i^N \frac{w(g^{\text{key}}, f_i^{\text{key}})(s)}{\sum_j^N w(g^{\text{key}}, f_j^{\text{key}})(s)} \cdot F_i^{\text{detail}}(\omega)(s) \quad ,$$

with the time-varying weighing w

$$w(f,g)(t) = \frac{1}{|f(t) - g(t)|} \quad ,$$

where $|\,|$ is a modified L_2 norm that weights orientation and position with differing coefficients. In practice, our scene scale allows us to use the same factor for both. When computing w, the border mode extends both f and g to be defined for every t, allowing for sequences of different length.

We then re-generate motion details according to the derived statistics $G^{\text{detail}}(\omega)(s)$. For this, we rely on uniformly distributed random numbers. These numbers are interpolated on varying scales and are used to sample according to the time-varying spectrum [38, 39]. We call $\mathcal{N}$ the operator that generates the signal according to the given spectrum:

$$g(t) = T(g^{\text{base}}(t)) \cdot T(\mathcal{N}(G^{\text{detail}}(\omega)(s))) \quad .$$

Note, that the details – by design – will only share the spectrum of what was observed and are not actual copies, i. e., they are non-repetitive.

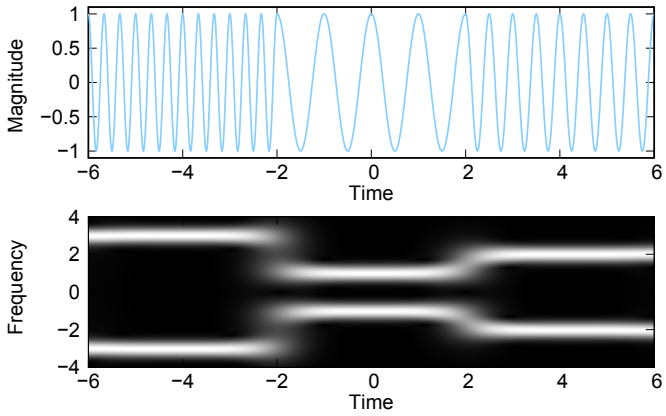

Figure 4: Example of a Gabor transform to perform a time-frequency analysis of a signal.

3.4 Application

In our system, a user loads a virtual scene that can be inspected in 3D in real-time. The user can define a camera motion path by placing splines in the scene. The system then synthesizes detail for that motion path interactively by querying the current path in the database of details.

Finally, we can hallucinate and synthesized motion frequencies that were not even present in the original 25 Hz motion observation. We do this by fitting a line in the spectral domain into the spectrum and extrapolate frequencies above the original data's cut-off frequency. When using a high-refresh-rate display (e. g. , 120 Hz), these supplementary details add further subtle movements that increase realism.

4 Results

While motion is difficult to reproduce on paper, depicting animation details is hardly possible and we refer to the accompanying video for a visualization.

For our experiments we generated a camera motion database with a total of 48 automatically analyzed sequences. These sequences are manually labeled with styles. We used 5 different labels: walking (17 sequences), running (2 sequences), bicycle (16 sequences), skateboard (9 sequences), and helicopter (4 sequences). Fig. 5 shows two example spectra extracted from a biking and a running sequence.

In a first example (please see the supplemental video), a user has set up a virtual scene (which is also shown in the teaser figure) and selects different style tags that alter the camera motion style. First, the user selects a simple walking motion from a certain movie as style. Our system transfers this walking style to a different camera animation. The result is a motion including a curve instead of the straight motion that was present in the original input movie. Furthermore, the motion is several times longer than the original input style example. The user then selects running-style motion details for a faster motion. Note that the speed changes the details. Then, the user selects a bicycle style, where mainly details in the y component of the position, as well as high-frequency rotational noise is present. Finally, when selecting the skateboard style, only high frequency positional noise remains.

In a second example (cf. Fig. 6), the user modifies a camera path, and details are synthesized interactively while the path changes. The details adapt to the shape of the path, i. e. , details orthogonal to the path remain orthogonal, even if the path is deformed. Finally, the users selects the strength and cutoff frequency of the details to reach the desired result.

Having captured the camera-shake spectrum, we made certain findings. We can confirm, that – similar to many things in nature – the power spectrum of camera shake is indeed $1/f^h$, with an $h \approx 2$, which is the same as, e. g. , for the intensity distribution in natural images.

5 Discussion

There are alternatives to our detail extraction approach. One option is to subtract a smoothed version of the path from the original path and directly rely on the difference. We did not use this approach, because the resulting animations are always repetitive and contain copied details (cp. Fig. 7).

The quality of the Gabor time-frequency resolution is limited, because we have to detail with a relatively limited sampling rate, compare to, e. g. , sound samples (25 Hz camera animation vs. typical 44 kHz audio).

Depending on the motion, our choice of 100 iterations for the Taubin filter might not be optimal. In practice, picking a higher smoothing will shift base information into the details, while doing less smoothing will only cause small parts of the signal to be detected as details. The details are finer for some paths, while they are coarser for others, and so choosing this parameter globally is a compromise. Nevertheless, due to our fixed input sampling rate, we found that our indicated global values make sense and behave well in most cases. In general, it would be interesting to find criteria that separate base and detail on a perceptual basis and for varying frame rates. Fig. 8 shows extracted details for different cut-off frequencies.

6 Conclusion

We presented a system to add plausible camera motion details to a virtual camera motion. The system synthesizes details at interactive rates, so novel camera animations can be readily inspected in real-time. We avoid repetitive structures in the motion and are able to mimic various styles.

In future work, we would like to find physical camera parameters (i. e. , weight, mass tensors), by applying inverse physical modeling to the extracted camera motion data. Once such data is available, novel camera paths could also take a physical model into account. Further, structuring the detail space according to spectral properties could be used to derive priors for other applications, i. e. , automatic classification of footage. In the

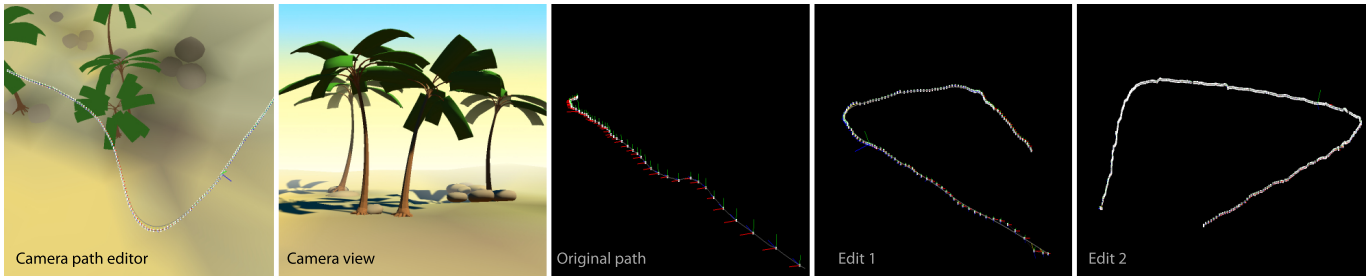

Figure 6: An edit session using our system. From left to right: Users manipulate a spline in a 3D world using an existing framework including an interactive camera preview. Our system adds the details from the style example to two edited path shapes. Note, how the edited path shapes are different from the style example (which is a straight line) in shape and duration, but the details are similar. An animated version of this figure can be found in the supplemental video.

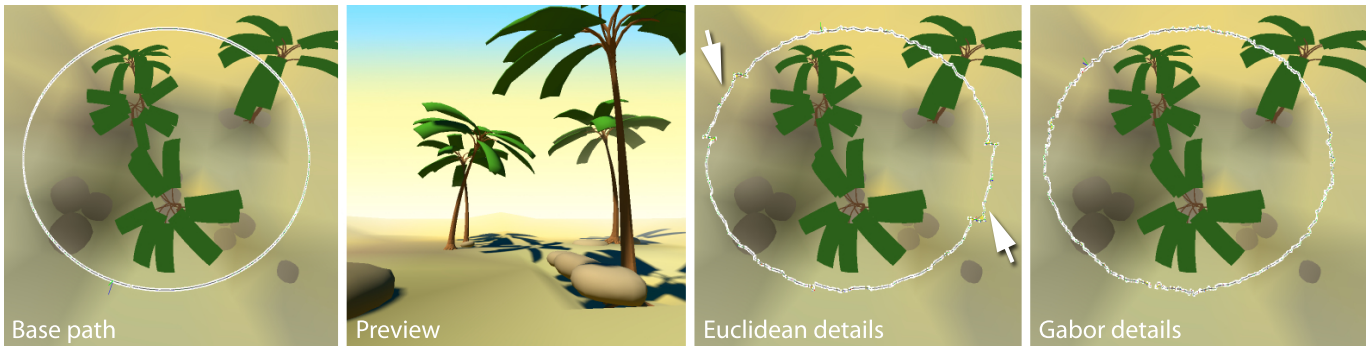

Figure 7: Synthesizing details from a time-varying spectrum instead of adding them directly avoids repetition. From left to right: Given a base path and a preview, adding details in Euclidean space can lead to repeating details (arrows), whereas Gabor details do not repeat and only share their spectral properties with the style example.

context of rendering, realistic motion blur [16] is achieved by convolving with the full point spread function (PSF) of the camera movement during exposure (cp. Fig. 1). We could use our spectrum extrapolation to generate intra-frame details from tracking data available only at frame resolution. The resulting motion blur would agree better with the expected spectrum of the camera movement. Finally, our results might also be of interest for stabilization, which could be easier, if the expected spectrum of camera shake can be modeled.

Acknowledgments

This work has been partially funded by the Max Planck Center for Visual Computing and Communication (BMBF-FKZ01IM10001) and the Intel Visual Computing Institute.

References

[1] L. Bacher, *Max Ophüls in the Hollywood Studios*. New Brunswick, NJ: Rutgers University Press, 1996.

[2] J. Blinn, "Where am I? What am I looking at?," *IEEE Comput. Graph. Appl.*, vol. 8, no. 4, pp. 76–81, 1988.

[3] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice in C*. Addison-Wesley Professional, 2nd ed., August 1995.

[4] M. Gleicher and A. Witkin, "Through-the-lens camera control," *SIGGRAPH Comput. Graph.*, vol. 26, no. 2, pp. 331–340, 1992.

[5] T. Kamada and S. Kawai, "A simple method for computing general position in displaying three-dimensional objects," *Comput. Vision Graph. Image Process.*, vol. 41, no. 1, pp. 43–56, 1988.

[6] S. M. Drucker, T. A. Galyean, and D. Zeltzer, "CINEMA: A system for procedural camera movements," in *Proceedings of the 1992 Symposium on Interactive 3D Graphics (I3D)*, (New York, NY, USA), pp. 67–70, ACM, 1992.

[7] L.-W. He, M. F. Cohen, and D. H. Salesin, "The virtual cinematographer: a paradigm for automatic real-time camera control and directing," in *SIGGRAPH'96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, (New York, NY, USA), pp. 217–224, ACM, 1996.

[8] W. Sack and M. Davis, "IDIC: Assembling video sequences from story plans and content annotations," in *Proc. Intl. Conf. on Multimedia Computing and Systems*, pp. 30–36, 1994.

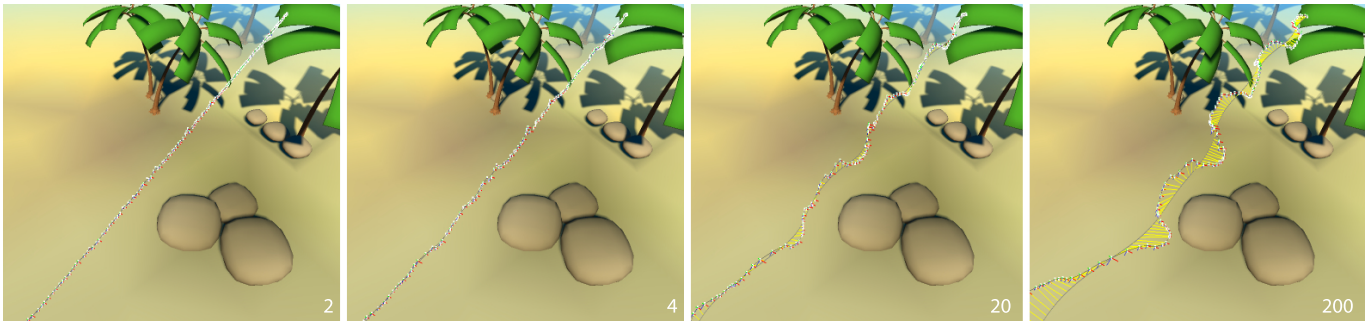

Figure 8: Details of different cut-off frequencies (in number of Taubin iterations). For very high values (200 and more for a 100 sample input style example), the complete style example is considered a detail and a straight line starts to look like the style example itself.

[9] R. Tsai, "An efficient and accurate camera calibration technique for 3-d machine vision," in *Proc. Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 364–374, 1986.

[10] P. E. Debevec and J. Malik, "Recovering high dynamic range radiance maps from photographs," in *SIGGRAPH'97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, (New York, NY, USA), pp. 369–378, 1997.

[11] S. Kuthirummal, A. Agarwala, D. B. Goldman, and S. K. Nayar, "Priors for large photo collections and what they reveal about cameras," in *ECCV '08: Proceedings of the 10th European Conference on Computer Vision*, (Berlin, Heidelberg), pp. 74–87, Springer-Verlag, 2008.

[12] C. Morimoto and R. Chellappa, "Evaluation of image stabilization algorithms," in *DARPA Image Understanding Workshop (DARPA97)*, pp. 295–302, 1997.

[13] C. Buehler, M. Bosse, and L. McMillan, "Non-metric image-based rendering for video stabilization," in *Proc. Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. II:609–614, 2001.

[14] F. Liu, M. Gleicher, H. Jin, and A. Agarwala, "Content-preserving warps for 3d video stabilization," *ACM Trans. Graph. (Proc. SIGGRAPH)*, vol. 28, no. 3, pp. 1–9, 2009.

[15] M. L. Gleicher and F. Liu, "Re-cinematography: improving the camera dynamics of casual video," in *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*, (New York, NY, USA), pp. 27–36, ACM, 2007.

[16] N. Joshi, S. B. Kang, C. L. Zitnick, and R. Szeliski, "Image deblurring using inertial measurement sensors," *ACM Trans. Graph. (Proc. SIGGRAPH)*, vol. 29, no. 4, pp. 1–9, 2010.

[17] M. Livingstone, *Vision and Art: The Biology of Seeing*. Harry N. Abrams, May 2002.

[18] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, "Image analogies," in *SIGGRAPH'01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, (New York, NY, USA), pp. 327–340, ACM, 2001.

[19] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE Comput. Graph. Appl.*, vol. 21, no. 5, pp. 34–41, 2001.

[20] S. Bae, S. Paris, and F. Durand, "Two-scale tone management for photographic look," *ACM Trans. Graph. (Proc. SIGGRAPH)*, vol. 25, no. 3, pp. 637–645, 2006.

[21] A. Fournier, D. Fussell, and L. Carpenter, "Computer rendering of stochastic models," *Commun. ACM*, vol. 25, no. 6, pp. 371–384, 1982.

[22] A. Golovinskiy, W. Matusik, H. Pfister, S. Rusinkiewicz, and T. Funkhouser, "A statistical model for synthesis of detailed facial geometry," *ACM Trans. Graph. (Proc. SIGGRAPH)*, vol. 25, no. 3, pp. 1025–1034, 2006.

[23] A. Bruderlin and L. Williams, "Motion signal processing," in *SIGGRAPH'95: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, (New York, NY, USA), pp. 97–104, ACM, 1995.

[24] M. Brand and A. Hertzmann, "Style machines," in *SIGGRAPH'00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, (New York, NY, USA), pp. 183–192, ACM, 2000.

[25] C. K. Liu, A. Hertzmann, and Z. Popović, "Learning physics-based motion style with nonlinear inverse optimization," *ACM Trans. Graph. (Proc. SIGGRAPH)*, vol. 24, no. 3, pp. 1071–1081, 2005.

[26] M. Gleicher, "Retargetting motion to new characters," in *SIGGRAPH'98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, (New York, NY, USA), pp. 33–42, ACM, 1998.

[27] J. Wang, S. M. Drucker, M. Agrawala, and M. F. Cohen, "The cartoon animation filter," *ACM Trans. Graph. (Proc. SIGGRAPH)*, vol. 25, no. 3, pp. 1169–1173, 2006.

[28] D. J. Heeger and J. R. Bergen, "Pyramid-based texture analysis/synthesis," in *SIGGRAPH'95: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, (New York, NY, USA), pp. 229–238, ACM, 1995.

[29] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, (Washington, DC, USA), p. 1033, IEEE Computer Society, 1999.

[30] L.-Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk, "State of the art in example-based texture synthesis," in *Eurographics 2009, State of the Art Report, EG-STAR*, Eurographics Association, 2009.

[31] M. Okabe, K.-i. Anjyo, T. Igarashi, and H.-P. Seidel, "Animating pictures of fluid using video examples," *Comput. Graph. Forum*, vol. 28, no. 2, pp. 677–686, 2009.

[32] T. Mertens, J. Kautz, J. Chen, P. Bekaert, and F. Durand, "Texture transfer using geometry correlation," in *Rendering Techniques 2006: 17th Eurographics Workshop on Rendering*, pp. 273–284, June 2006.

[33] T. Thormählen and H. Broszio, "Voodoo camera tracker." free download at http://www.digilab.uni-hannover.de.

[34] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch, "Visual modeling with a hand-held camera," *Int. J. Comput. Vision*, vol. 59, no. 3, pp. 207–232, 2004.

[35] S. Gibson, J. Cook, T. Howard, R. Hubbold, and D. Oram, "Accurate camera calibration for off-line, video-based augmented reality," in *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2002)*, (Darmstadt, Germany), 2002.

[36] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.

[37] G. Taubin, "A signal processing approach to fair surface design," in *SIGGRAPH'95: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, (New York, NY, USA), pp. 351–358, ACM, 1995.

[38] K. Perlin, "An image synthesizer," in *SIGGRAPH'85: Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, (New York, NY, USA), pp. 287–296, ACM, 1985.

[39] A. Lagae, S. Lefebvre, R. Cook, T. DeRose, G. Drettakis, D. Ebert, J. Lewis, K. Perlin, and M. Zwicker, "State of the art in procedural noise functions," in *EG 2010 - State of the Art Reports* (H. Hauser and E. Reinhard, eds.), Eurographics, Eurographics Association, May 2010.