Coherent Noise for Non-Photorealistic Rendering

Michael Kass* Davide Pesare †

Pixar Animation Studios

Abstract

A wide variety of non-photorealistic rendering techniques make use of random variation in the placement or appearance of primitives. In order to avoid the "shower-door" effect, this random variation should move with the objects in the scene. Here we present *coherent noise* tailored to this purpose. We compute the coherent noise with a specialized filter that uses the depth and velocity fields of a source sequence. The computation is fast and suitable for interactive applications like games.

CR Categories: I.3.3 [Computer Graphics]—Picture/Image Generation; I.4.3 [Image Processing and Computer Vision]—contrast enhancement, filtering

Keywords: Non-photorealistic rendering, noise, painterly rendering

1 Introduction

In 1985, Peachey [1985] and Perlin [1985] simultaneously introduced the idea of using procedural noise for solid texturing. Since then, the method has been refined (e.g. [Cook and DeRose 2005; Lagae et al. 2009]) to provide greater control over the spectral characteristics of the noise and has become an essential tool in photorealistic rendering. The demands of non-photorealistic rendering, however, are different enough that existing noise techniques fail to address some important issues.

While non-photorealistic rendering is a wide and heterogeneous field, many of the important applications for random variation have common requirements. Styles derived from hand painting and drawing tend to need relatively uniform 2D spectral properties in the image plane to achieve a unity of composition and style. Nonetheless, the random variations must track the movement of objects to avoid the well-known *shower door* effect, the illusion that the random variation exists on a piece of glass through which the scene is being viewed. None of the traditional techniques for generating noise are well-suited to these requirements. Solid or surface noise attached to objects in 3D will have non-uniform 2D spectra. Noise generated in the image plane will generally not track the motion of the objects in the scene.

Here we introduce a new approach for computing random variation with the needed characteristics. We begin with rendered depth and velocity fields. We then take a block of white noise as a function of (x, y, t) and filter it, taking into account the depth and velocity fields and their consequent occlusion relationships. The result is what we call a *coherent noise field*. Each frame alone looks like independent white noise, but the variation from frame to frame is consistent with the movement in the scene. The resulting noise can be queried by non-photorealistic rendering algorithms to create random variation with uniform image-plane spatial properties that nonetheless appear firmly attached to the 2d projections of the 3D objects.

2 Previous Work

Bousseau et al. [2007] developed a technique for watercolor stylization that can be used for very similar purposes to the present work. Their method is based on the idea of advecting texture coordinates. To initialize it, texture coordinates are assigned to each pixel on the first frame of a sequence using an undistorted rectilinear map. From frame to frame, the texture coordinates are advected based on the image-space velocity of each pixel. If these texture coordinates are used to index into a noise texture, the noise will move with the objects in the scene.

The difficulty with the prior work on advecting texture coordinates [Max and Becker 1995; Neyret 2003] is that as a sequence progresses, the mapping implied by the texture coordinates become more and more distorted, and the fill-in at disoccluded regions becomes problematic. As a result, the texture-mapped noise starts to acquire non-stationary spatial frequencies. Areas stretched out by the mapping will become blurry, and compressed areas will display higher spatial frequencies than in the original noise.

Bousseau et al. provide a solution to this problem, although it comes with a key limitation. They divide each sequence into blocks of frames. For each block, they compute two sets of advected texture coordinates. One set of coordinates is computed as before, starting at the first frame in the block and advecting forward through time. The other set of coordinates is initialized on the last frame in the block, and advected backwards in time. Noise mapped through the first set of coordinates gets more and more distorted as time progresses. Noise mapped through the second set becomes less and less distorted. With a suitable blend of the two mapped noise functions, Bousseau et al. achieve noise that appears relatively stationary.

The key limitation of this approach is that it requires knowledge about the future. For offline rendering of non-photorealistic animation, this is not a problem. Information about the future of each frame can be made available at rendering time. For interactive or real-time applications, however, this information is not available, and the method cannot be used. For these types of applications, we offer our coherent noise instead.

In concurrent work, Benard et al. [2010] propose a method based on Gabor noise. They splat Gabor kernels around a set of seed points on 3D models. Since their seed points are fixed to the 3D model, the method can generate directional noise with a direction fixed to the 3D surface. Our image-space method lacks any fixed 3D reference, so it does not have this capability.

The kernel splat used by Benard et al., however, does not respect visibility. When a seed point is visible, the entire kernel is splatted.

*e-mail: kass@pixar.com

†e-mail: dakrunch@pixar.com

ACM Reference Format
Kass, M., Pesare, D. 2011. Coherent Noise for Non-Photorealistic Rendering.
ACM Trans. Graph. 30, 4, Article 30 (July 2011), 5 pages. DOI = 10.1145/1964921.1964925
http://doi.acm.org/10.1145/1964921.1964925.

As a result, popping artifacts are possible as the visibility of seed points changes. By contrast, our method makes use of explicit visibility calculations for each component of the calculation and avoids popping artifacts.

3 Filtered White Noise

In fully synthetic 3D computer graphics applications, we can generally expect to be able to extract additional information at each pixel besides the rendered color vector. In particular, let us assume that the depth $z_i(x,y)$ and the 3D velocity vector $V_i(x,y)$ are available for each frame i, $1 \leq i \leq n$ in a given shot. This information is available from popular commercial renderers and is easy to generate with OpenGL and other interactive rendering libraries. Our goal is to use the information to create a coherent noise field $c_i(x,y)$ with stationary statistics in each frame, but with correlations between frames that match the motion of the scene.

We begin by creating a block of independent, identically distributed white noise $w_i(x,y)$ for all $1 \leq i \leq n$. We then create the necessary correlations by filtering the noise from frame to frame. In order to do the low-pass filtering efficiently, we use a discrete time recursive filter [Oppenheim and Schafer 1975].

The simplest recursive filter suffices for our purposes. If the input to the filter is f_i and the output of the filter is g_i, a first-order recursive filter can be written

$$g_i = \alpha g_{i-1} + (1-\alpha) f_i. \tag{1}$$

where α controls the exponential rate of decay of the filter's impulse response given by

$$h_i = \begin{cases} (1-\alpha)\alpha^i & i \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

and depicted in Figure 1.

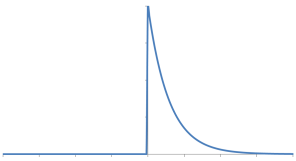


Figure 1: *First-order recursive filter impulse response*

3.1 Stationarity

In order to create coherent noise that is easy to use, we need to ensure that the the output of our filter is stationary – that its statistics do not depend on the frame number i. We can do this by setting appropriate conditions on g_0, which is left unspecified by the recursion in Equation 1.

In our case, the quantity corresponding to f_i that we are filtering is independent, identically distributed noise. Thus we can write the variance of each side of Equation 1 as follows:

$$\sigma^2(g_i) = \alpha^2\sigma^2(g_{i-1}) + (1-\alpha)^2\sigma^2(f) \tag{3}$$

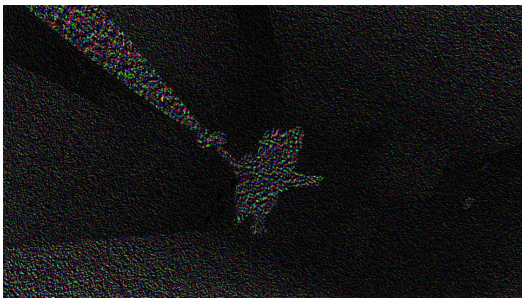
The condition we would like is $\sigma^2(g_i) = \sigma^2(g_{i-1})$ which implies

$$\sigma^2(g) = \alpha^2\sigma^2(g) + (1-\alpha)^2\sigma^2(f) \tag{4}$$

(a) Original frame of animation.

(b) Frame one of coherent noise.

(c) Frame two of coherent noise.



(d) The difference between (b) and (c).

Figure 2: *Coherent noise.*

Some simple algebra yields

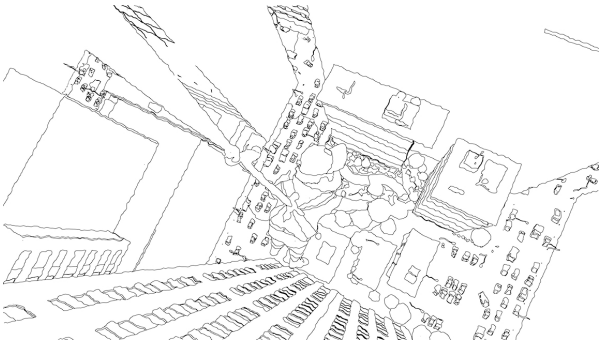
$$\sigma^2(g) = \left(\frac{1-\alpha}{1+\alpha}\right)\sigma^2(f) \tag{5}$$

which gives us the necessary condition for stationarity

$$g_0 = \sqrt{\frac{1-\alpha}{1+\alpha}}\, f_0 = k(\alpha)\, f_0. \tag{6}$$

Substitution back in Equation 3 shows that the condition is also sufficient.

(a) Using dilate/erode/warp to stylize.



(b) Perturbing object-ID boundaries.

Figure 3: *Applications of coherent noise.*

3.2 Disocclusion

Having determined the condition for stationarity, we can return to our goal of filtering white noise images $w_i(x,y)$ into coherent noise $c_i(x,y)$. Initially, we set $c_0(x,y) = k(\alpha)w_0(x,y)$ for each pixel (x,y). Then we compute $c_i(x,y)$ from $c_{i-1}(x,y)$ as follows.

We know that the front-most surface at pixel (x,y) had depth $z_i(x,y)$ and velocity $V_i(x,y)$ at frame i. Knowing the time interval Δt between frames, we can estimate the 3D position of that surface point at the previous frame: $(x',y',z') = (x,y,z) - \Delta t V_i(x,y)$. We then look at the depth $z_{i-1}(x',y')$ and compare it to z'. If z' is farther away than $z_{i-1}(x',y')$ by at least some minimum amount ε, we can conclude that the surface point was previously occluded by a closer surface at frame $i-1$. Otherwise, we conclude that the point was previously visible. Note that since (x',y') are not necessarily integer, computing $z_{i-1}(x',y')$ will generally require interpolation. We use bilinear interpolation for this purpose.

Having determined whether or not the surface point at (x,y) has just become disoccluded, we compute $c_i(x,y)$ as follows:

$$c_i(x,y) = \begin{cases} k(\alpha)w_i(x,y) & \text{disocclusion} \\ \alpha c_{i-1}(x',y') + (1-\alpha)w_i(x,y) & \text{otherwise} \end{cases} \quad (7)$$

Here too, we compute $c_{i-1}(x',y')$ using bilinear interpolation.

3.3 Time-Symmetric Filtering

The filter described in section 3.2 is causal–the output at a given frame depends only on information that precedes that frame. This is a vital property for applications like games, where the future is unknown. For offline rendering of animation, however, the future is known, and a higher quality result can be created by using a symmetric filter which looks into the future as much as it looks into the past. To achieve this, we can do the filtering both forward and backwards in time.

In many cases, time-symmetric filtering can be achieved by cascading two identical recursive filters. The output of the forward filter is fed to the input of the backwards filter. In this case, however, if we took the output of the forward filtering and then filtered it backwards, the input would no longer be uncorrelated white nose, and the previous conditions for stationarity would not hold. Instead, we separately filter the same white noise forwards and backwards in time and add the result.

There remains one subtle point. Let $\underline{c}_i(x,y)$ be the result of filtering $w_i(x,y)$ as described, and let $\overline{c}_i(x,y)$ be the result of the same process done with time reversed, starting at frame n and ending at frame i. Then the sum double counts at the center of its impulse response.

If we use the recursive filter of Equation 1 forward and backward in time

$$\begin{array}{rcl} \underline{g}_i & = & \alpha \overline{g}_{i-1} + (1-\alpha) f_i \\ \overline{g}_{i-1} & = & \alpha \underline{g}_i + (1-\alpha) f_{i-1} \end{array} \quad (8)$$

then the sum has impulse response

$$h_i = \begin{cases} (1-\alpha)\alpha^i & i \neq 0 \\ 2(1-\alpha) & i = 0 \end{cases} \quad (9)$$

because the impulse responses of the forward and backward filters overlap at offset zero. The result is a poor low-pass filter. To fix this, we can simply subtract $(1-\alpha)f_i$ from the sum producing a filter with impulse response

$$h_i = (1-\alpha)\alpha^{|i|} \quad (10)$$

shown in Figure 4.

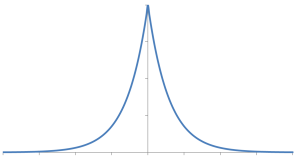


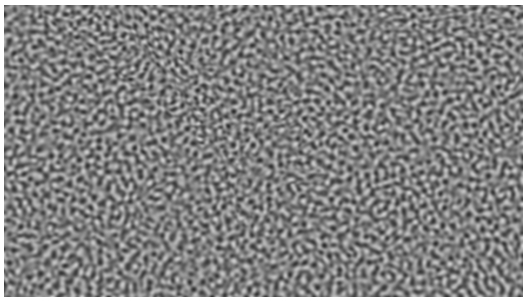
Figure 4: *Symmetric first-order recursive filter impulse response*

The full algorithm is to use the computation forward in time to compute $\underline{c}_i(x,y)$ and then repeat the computation backwards in time to compute $\overline{c}_i(x,y)$. Finally, the result is given by

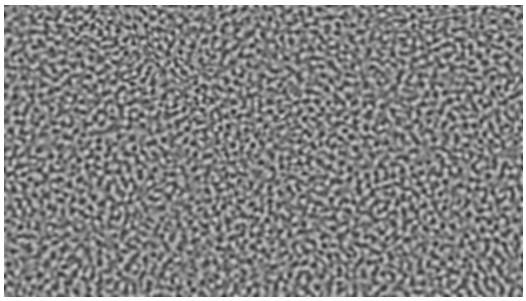
$$\mathcal{C}_i(x,y) = \overline{c}_i(x,y) + \underline{c}_i(x,y) - (1-\alpha)w_i(x,y). \quad (11)$$

3.4 Spatial Spectrum

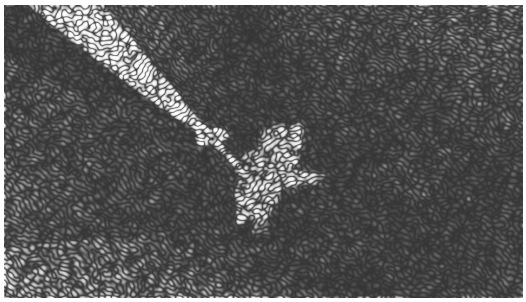
Some control over the spectrum of $\mathcal{C}$ can be achieved through filtering. Instead of white noise $w_i(x,y)$, we can substitute noise with a more appropriate spatial spectrum as the input to our recursive filtering. As long as the individual frames of noise are independent from each other, the same conditions for stationarity of the variance will still hold. If we want some continuity across object boundaries, we can also do a small amount of spatial smoothing of the coherent noise, whether computed in a causal or time-symmetric manner. Figure 5 shows the result of pre-filtering with a spatial band-pass filter. Once again, each frame looks spatially uniform, while the structure of the object motion is visible in the difference between frames, here presented in absolute value.



(a) Frame one of bandpass coherent noise.



(b) Frame two of bandpass coherent noise.



(c) The absolute value of the difference between (b) and (c).

Figure 5: *Bandpass coherent noise.*

3.5 Limitations

The parameter α controls the width of the impulse response and hence the time period over which noise becomes fully refreshed. For good results, there are limits to this time period. If the chosen period is excessively long relative to the speed of the motion, extreme deformations of the noise during the period can create non-uniform spatial characteristics in the noise. Fast translations and rotations in the image plane are not problematic, but motion that causes extremely rapid changes in projected 2D area can degrade the quality of the noise. For applications where this is a problem, the technique of Neyret [2003] can be used to blend noise fields calculated with different values of α.

Note that nothing in this method is particularly well-suited to highly directional noise. Unlike the method of Benard et al. [2010], the filtering proposed here does not keep track of the 3D information necessary to orient directional noise consistently with respect to a 3D model.

4 Results

Figure 2 shows the results for a sequence from the Disney/Pixar animated feature "Up". We used a small amount of Gaussian smoothing of the input noise and a small amount of Gaussian smoothing of the result. We repeated the calculation independently for each color channel and used $\alpha = .98$.

A frame from the original animation is shown in Figure 2(a). The images in Figure 2(b) and Figure 2(c) are two frames from the coherent noise computed for this shot. On casual inspection, they appear to be unrelated noise. Subtracting them, however, yields the image in Figure 2(c) which clearly shows the time-coherent structure and reveals the character dangling by a make-shift rope.

On way to visualize the operation of our filter is to look at what happens to a single scan line over time. In figure 6(a), we have selected a scan line from the input image. In figure 6(b), we show that same scan line over time, where time is represented on the vertical axis. The bottom row of pixels of figure 6(b) correspond to the frame shown in figure 6(a). Each row of pixels above it shows that same scan line at a later time. In figure 6(c), we have created the same display of our causal coherent noise through time. Note that figure 6(c) resembles a line-integral convolution [Cabral and Leedom 1993] visualization of the velocity field. Low variation along the velocity field in this picture is another way to express temporal coherence of the noise field when tracking an individual object point.

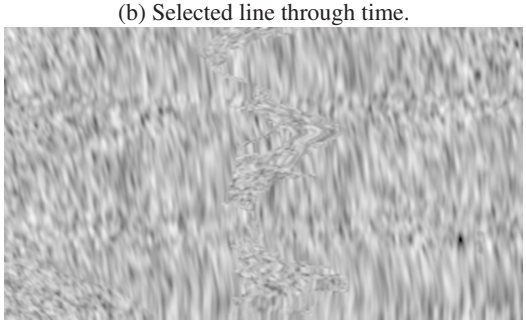
The images in Figure 2 are stills from motion sequences on the accompanying video. The coherent noise is shown first for the "one pass" causal computation described in section 3.2 and then for the time-symmetric "double-pass" computation described in section 3.3. Note that the three-dimensional shapes in the original example are clearly evident despite the fact that each frame by itself lacks any 3D structure.

Figure 3(a) shows the first example of the use of coherent noise for a non-photorealistic rendering effect. We have taken the original geometry of the scene shown in Figure 2(a) and re-rendered it with a single distant light source, without shadows and with quantized shading. Then we added spatially smoothed coherent noise and performed a dilation and an erosion step [Haralick et al. 1987] to create the texture. Finally, we used two channels of coherent noise to displace the image with a local image-plane warp. All the 2D operations were performed in Apple's *Shake* compositing software. The "ColorSplat" example in the accompanying video shows the result in motion. First, the video shows the result with white noise. Each sample of the noise is generated as an independent random variable. The result is chaotic and difficult to watch. Next, the video shows the result of using what we call constant noise. This is noise that is a function of image spce, but which does not change with time. With constant noise, the shower-door effect is very evident – it looks as if the texture exists on a piece of glass through which the scene is being viewed. Finally, the video shows the use of our causal (one pass) NPR noise. In this case, the random variations due to the noise accurately track the motion of the object.

Our second example "edge detect" is based on modifying a set of edges. Here we have rendered object ID's at each pixel and detected the boundaries where the object ID's change. We then chained the boundaries together into curves. The first video segment shows the curves unmodified by noise. Unfortunately there is some aliasing due to the extreme geometric complexity of the scene. For each point on each curve, we compute a 2D displacement from two channels of noise. We evaluate the noise at a point slightly displaced toward the side of the curve which is closest to the observer, so the noise will be associated with the front-most object. Finally, we low-pass filter the noise along the length of the curve, and use it as

(a) Selected line of original

(b) Selected line through time.



(c) Same line of noise through time.

Figure 6: *Relationship of NPR noise to line-integral convolution.*

a screen-space displacement for each vertex. One again, the white noise flashes and the constant noise produces a strong shower-door effect. With our NPR noise, the wiggles on the curves move nicely and coherently with the motion of the objects in the scene. If full temporal consistency is not desired, for example to simulate some of the effect of hand-drawn animation, a lower value for α can be chosen. In the limit as α goes to zero, the coherent noise will turn into independent noise. Note that the wiggles in our line rendering have uniform 2D size and are independent of distance, object or parameterization.

The use of coherent noise with uniform 2D spatial properties does not prevent varying the spatial properties based on artistic intent. As with Perlin noise, our coherent noise can be calculated at different spatial scales and combined together procedurally to achieve desired artistic goals.

5 Conclusion

Perlin noise and its refinements have shown the clear value of noise as a simple primitive building block on which to build a variety of interesting effects. We offer our coherent noise in the same spirit. Like Perlin noise, it is easy to implement, and inexpensive to compute. Unlike Perlin noise, it tracks two-dimensional motion while maintaining stationary spatial properties in screen space. Also, unlike the approach of Bousseau et al., it works well in contexts like games and other interactive applications where the future is unknown. We hope that the effectiveness, ease of implementation and wide applicability of this coherent noise technique will make it an attractive choice for future work in non-photorealistic rendering.

While we have presented results for our filter operating on incoherent white or band-pass noise, the technique is potentially more widely applicable to situations where one wants improve the temporal coherence of an image sequence which is nearly temporally coherent, but not quite coherent enough. Any of a variety of NPR techniques which occasionally pop or suffer from other temporal artifacts could gain improved quality from our velocity- and occlusion-aware filtering. For example, where directional noise with the highest quality is desired, the noise of Benard et al. could be further processed with our filtering method to improve its temporal coherence.

Acknowledgements

We thank Pixar Animation Studios for permission to use data from the film “Up” to illustrate our technique. We thank one of the anonymous reviewers for pointing out the relationship of this work to line-integral convolution.

References

BÉNARD, P., LAGAE, A., VANGORP, P., LEFEBVRE, S., DRETTAKIS, G., AND THOLLOT, J. 2010. A dynamic noise primitive for coherent stylization. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 29*, 4 (June).

BOUSSEAU, A., NEYRET, F., THOLLOT, J., AND SALESIN, D. 2007. Video watercolorization using bidirectional texture advection. *ACM Trans. Graph. 26*, 3, 104.

CABRAL, B., AND LEEDOM, L. C. 1993. Imaging vector fields using line integral convolution. In *SIGGRAPH*, 263–270.

COOK, R. L., AND DEROSE, T. 2005. Wavelet noise. *ACM Trans. Graph. 24*, 3, 803–811.

HARALICK, R., STERNBERG, S., AND ZHUANG, X. 1987. Image analysis using mathematical morphology. *PAMI 9*, 4 (July), 532–550.

LAGAE, A., LEFEBVRE, S., DRETTAKIS, G., AND DUTRÉ, P. 2009. Procedural noise using sparse gabor convolution. In *SIGGRAPH ’09: ACM SIGGRAPH 2009 papers*, 1–10.

MAX, N., AND BECKER, B. 1995. Flow visualization using moving textures. In *Proceedings of the ICAS/LaRC Symposium on Visualizing Time-Varying Data*, 77–87.

NEYRET, F. 2003. Advected textures. In *ACM-SIGGRAPH/EG Symposium on Computer Animation (SCA)*.

OPPENHEIM, A. V., AND SCHAFER, R. W. 1975. *Digital Signal Processing*. Prentice–Hall.

PEACHEY, D. R. 1985. Solid texturing of complex surfaces. In *SIGGRAPH*, 279–286.

PERLIN, K. 1985. An image synthesizer. In *SIGGRAPH*, 287–296.