Eurographics Symposium on Rendering 2010
Jason Lawrence and Marc Stamminger
(Guest Editors)

Volume 29 (2010), Number 4

A Dynamic Noise Primitive for Coherent Stylization
Styles Cookbook

P. Bénard[1] A. Lagae[2,3] P. Vangorp[3] S. Lefebvre[4] G. Drettakis[3] J. Thollot[1]

[1]Grenoble University [2]Katholieke Universiteit Leuven [3]INRIA Sophia-Antipolis [4]INRIA Nancy Grand-Est / Loria

1. Rendering Algorithm for Temporally Coherent Stylization

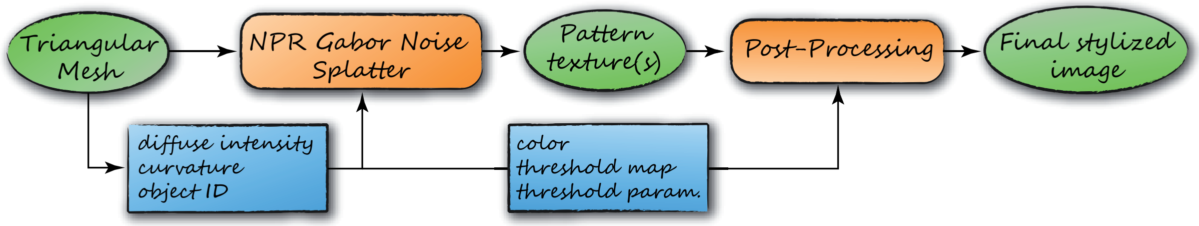


Figure 1: *Schematic representation of our rendering algorithm for temporally coherent stylization using NPR Gabor Noise.*

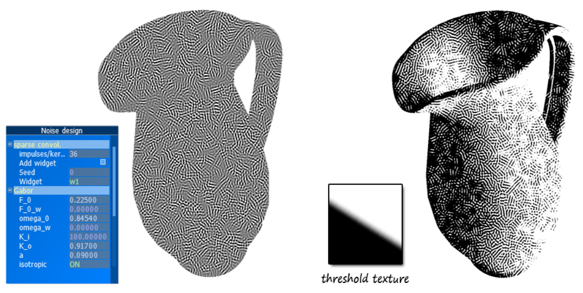
Our rendering algorithm for temporally coherent stylization is a two-step process (see figure 1) based on deferred shading. In the first step, we generate one or more noise layers using NPR Gabor noise. This is done using a general noise shader. We can use scene attributes (such as shading, surface curvature [Rus04] and object ID) to locally control the parameters of the noise (frequency, bandwidth and orientation). In the second step, we composite the noise layers to produce the final result. This is done using a style-specific shader.

In the remainder of this document, we give a detailed description of the individual styles (Sec. 2), including the threshold function we used for several styles (Sec. 3), and the compositing functions we used (Sec. 4).

2. Styles

In this section we provide pseudo-code for the style-specific shaders as well as the parameters that we used to produce the styles shown in the paper.

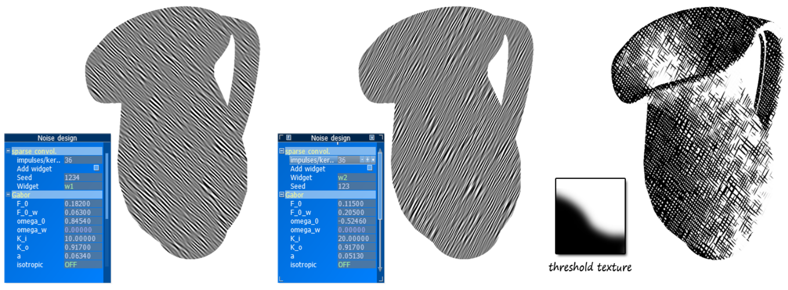
2.1. Stippling



```
Noise design
- sparse convol.
    impulses/ker..  36
    Add widget
    Seed            0
    Widget          w1
- Gabor
    F_0             0.22500
    F_0_w           0.00000
    omega_0         0.84540
    omega_w         0.00000
    K_i             100.00000
    K_o             0.91700
    a               0.09000
    isotropic       ON
```

```
float pattern = texture(noise_texture0, texCoord).r;
vec3  stroke  = texture(thresholdMap,vec2(pattern1,diffuseIntensity)).rgb;

vec3  final_color = stroke;
```

2.2. Cross-Hatching

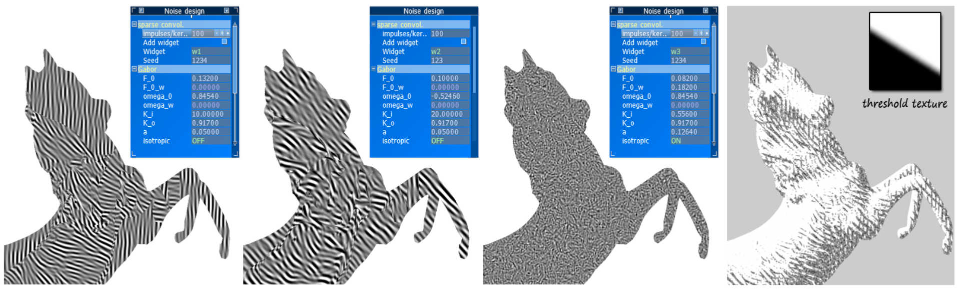


```
Noise design
sparse convol.
  impulses/ker..  36
  Add widget
  Seed            1234
  Widget          w1
Gabor
  F_0             0.18200
  F_0_w           0.06300
  omega_0         0.84540
  omega_w         0.00000
  K_i             10.00000
  K_o             0.91700
  a               0.06340
  isotropic       OFF
```

```
Noise design
sparse convol.
  impulses/ker..  36
  Add widget
  Widget          w2
  Seed            123
Gabor
  F_0             0.11500
  F_0_w           0.20500
  omega_0         -0.52460
  omega_w         0.00000
  K_i             20.00000
  K_o             0.91700
  a               0.05130
  isotropic       OFF
```

```
vec3 stroke1 = texture(thresholdMap,vec2(pattern1,diffuseIntensity)).rgb;
vec3 stroke2 = texture(thresholdMap,vec2(pattern2,diffuseIntensity)).rgb;

vec3 final_color = stroke1*stroke2;
```

2.3. Graphite



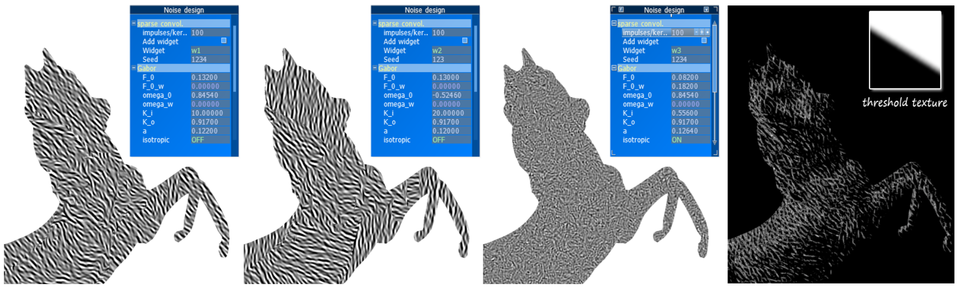
```
Noise design
sparse convol.
  impulses/ker..  100
  Add widget
  Widget          w1
  Seed            1234
Gabor
  F_0             0.13200
  F_0_w           0.00000
  omega_0         0.84540
  omega_w         0.00000
  K_i             10.00000
  K_o             0.91700
  a               0.05000
  isotropic       OFF
```

```
Noise design
sparse convol.
  impulses/ker..  100
  Add widget
  Widget          w2
  Seed            123
Gabor
  F_0             0.10000
  F_0_w           0.00000
  omega_0         -0.52460
  omega_w         0.00000
  K_i             20.00000
  K_o             0.91700
  a               0.05000
  isotropic       OFF
```

```
Noise design
sparse convol.
  impulses/ker..  100
  Add widget
  Widget          w3
  Seed            1234
Gabor
  F_0             0.08200
  F_0_w           0.18200
  omega_0         0.84540
  omega_w         0.00000
  K_i             0.55600
  K_o             0.91700
  a               0.12640
  isotropic       ON
```

```
float pattern1 = texture(noise_texture0, texCoord).r;
float pattern2 = texture(noise_texture1, texCoord).r;
float pattern3 = texture(noise_texture2, texCoord).r;

vec3  stroke1 = texture(thresholdMap,vec2((pattern1+pattern3)/2.0,diffuseIntensity)).rgb;
vec3  stroke2 = texture(thresholdMap,vec2((pattern2+pattern3)/2.0,diffuseIntensity)).rgb;

vec3  final_color = (1.0-(1.0-stroke1*stroke2)*pattern3);

if(isBackground) final_color = vec3(0.8);
```

2.4. Chalk



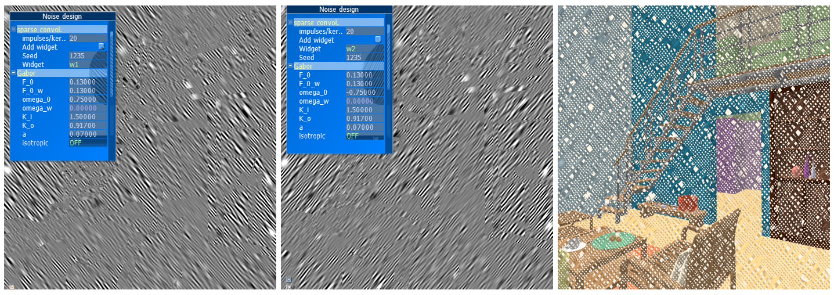
```
Noise design
- sparse convol.
  impulses/ker..  100
  Add widget
  Widget          w1
  Seed            1234
- Gabor
  F_0             0.13200
  F_0_w           0.00000
  omega_0         0.84540
  omega_w         0.00000
  K_i             10.00000
  K_o             0.91700
  a               0.12200
  isotropic       OFF
```

```
Noise design
- sparse convol.
  impulses/ker..  100
  Add widget
  Widget          w2
  Seed            123
- Gabor
  F_0             0.13000
  F_0_w           0.00000
  omega_0         -0.52460
  omega_w         0.00000
  K_i             20.00000
  K_o             0.91700
  a               0.12000
  isotropic       OFF
```

```
Noise design
- sparse convol.
  impulses/ker..  100
  Add widget
  Widget          w3
  Seed            1234
- Gabor
  F_0             0.08200
  F_0_w           0.18200
  omega_0         0.84540
  omega_w         0.00000
  K_i             0.55600
  K_o             0.91700
  a               0.12640
  isotropic       ON
```

```
float pattern1 = texture(noise_texture0, texCoord).r;
float pattern2 = texture(noise_texture1, texCoord).r;
float pattern3 = texture(noise_texture2, texCoord).r;

vec3 stroke1 = texture(thresholdMap,vec2((pattern1+pattern3)/2.0,diffuseIntensity)).rgb;
vec3 stroke2 = texture(thresholdMap,vec2((pattern2+pattern3)/2.0,diffuseIntensity)).rgb;

vec3 final_color = 1.0-(1.0-(1.0-stroke1*stroke2)*pattern3);

if(isBackground) final_color = vec3(0.0);
```

2.5. Color Strokes

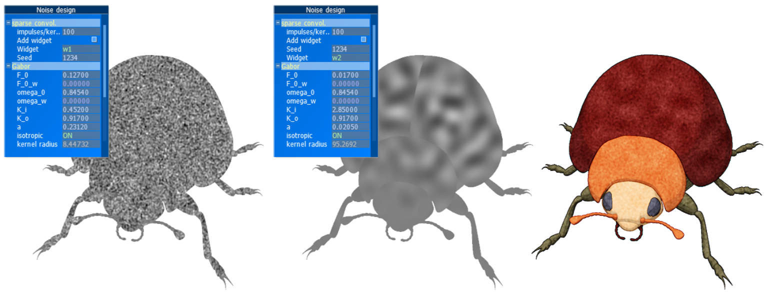


```
Noise design
- sparse convol.
  impulses/ker..  20
  Add widget
  Seed            1235
  Widget          w1
- Gabor
  F_0             0.13000
  F_0_w           0.13000
  omega_0         0.75000
  omega_w         0.00000
  K_i             1.50000
  K_o             0.91700
  a               0.07000
  isotropic       OFF
```

```
Noise design
- sparse convol.
  impulses/ker..  20
  Add widget
  Widget          w2
  Seed            1235
- Gabor
  F_0             0.13000
  F_0_w           0.13000
  omega_0         -0.75000
  omega_w         0.00000
  K_i             1.50000
  K_o             0.91700
  a               0.07000
  isotropic       OFF
```

```
float pattern1 = texture2D(noise_texture0, texCoord).r;
float pattern2 = texture2D(noise_texture1, texCoord).r;

vec3 background = vec3(0.99,0.96,0.9);
vec3 final_color = background;

float alpha = threshold(pattern1, 25, 0.5);
final_color = alphaBlend(final_color, overlay(color*0.85,pattern1,1.2), alpha);

alpha = threshold(pattern2, 25, 0.5);
final_color = alphaBlend(final_color, overlay(color,pattern2,1.0), alpha);

if(isBackground) final_color = background;
```

2.6. Watercolor

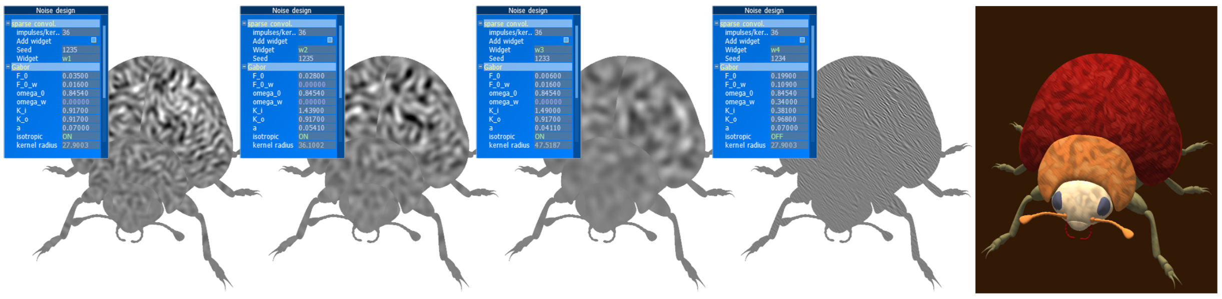


```
Noise design
- sparse convol.
  impulses/ker..  100
  Add widget
  Widget          w1
  Seed            1234
- Gabor
  F_0             0.12700
  F_0_w           0.00000
  omega_0         0.84540
  omega_w         0.00000
  K_i             0.45200
  K_o             0.91700
  a               0.23120
  isotropic       ON
  kernel radius   8.44732
```

```
Noise design
- sparse convol.
  impulses/ker..  100
  Add widget
  Seed            1234
  Widget          w2
- Gabor
  F_0             0.01700
  F_0_w           0.00000
  omega_0         0.84540
  omega_w         0.00000
  K_i             2.85000
  K_o             0.91700
  a               0.02050
  isotropic       ON
  kernel radius   95.2692
```

We use a watercolor stylization algorithm similar to the algorithm by Bousseau *et al.* [BKTS06], using one turbulence texture and one pigment texture.

2.7. Ink on Canvas



```
Noise design
- sparse convol.
  impulses/ker..  36
  Add widget
  Seed            1235
  Widget          w1
- Gabor
  F_0             0.03500
  F_0_w           0.01600
  omega_0         0.84540
  omega_w         0.00000
  K_i             0.91700
  K_o             0.91700
  a               0.07000
  isotropic       ON
  kernel radius   27.9003
```

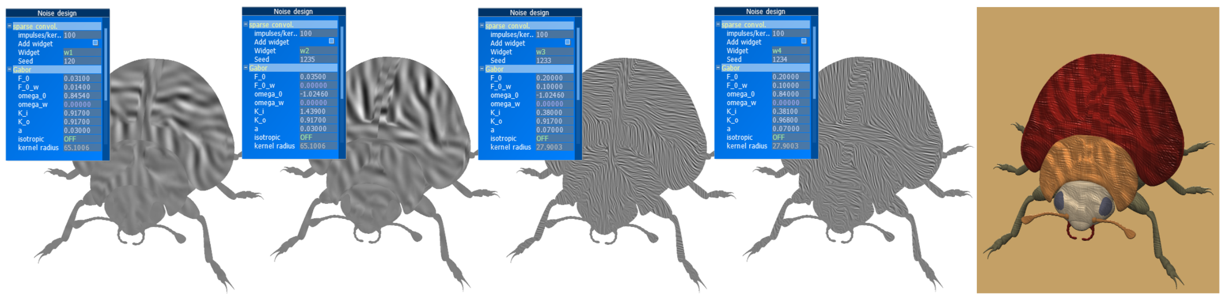
```
Noise design
- sparse convol.
  impulses/ker..  36
  Add widget
  Widget          w2
  Seed            1235
- Gabor
  F_0             0.02800
  F_0_w           0.00000
  omega_0         0.84540
  omega_w         0.00000
  K_i             1.43900
  K_o             0.91700
  a               0.05410
  isotropic       ON
  kernel radius   36.1002
```

```
Noise design
- sparse convol.
  impulses/ker..  36
  Add widget
  Widget          w3
  Seed            1233
- Gabor
  F_0             0.00600
  F_0_w           0.01600
  omega_0         0.84540
  omega_w         0.00000
  K_i             1.49000
  K_o             0.91700
  a               0.04110
  isotropic       ON
  kernel radius   47.5187
```

```
Noise design
- sparse convol.
  impulses/ker..  36
  Add widget
  Widget          w4
  Seed            1234
- Gabor
  F_0             0.19900
  F_0_w           0.10900
  omega_0         0.84540
  omega_w         0.34000
  K_i             0.38100
  K_o             0.96800
  a               0.07000
  isotropic       OFF
  kernel radius   27.9003
```

```
float pattern1 = texture2D(noise_texture0, texCoord).r;
float pattern2 = texture2D(noise_texture1, texCoord).r;
float pattern3 = texture2D(noise_texture2, texCoord).r;
float pattern4 = texture2D(noise_texture3, texCoord).r;

vec3  final_color = color;

float alpha = threshold(pattern3, 25, 0.5);
final_color = alphaBlend(final_color, overlay(color*0.9,pattern3,1.0), alpha);

alpha = threshold(pattern2, 25, 0.5);
final_color = alphaBlend(final_color, overlay(color*0.95,pattern2,1.0), alpha);

alpha = threshold(pattern1, 25, 0.5);
final_color = alphaBlend(final_color, overlay(color*1.05,pattern1,1.0), alpha);

final_color = overlay(final_color,pattern4,1.2);

if(isBackground) final_color = vec3(0.17,0.1,0.017);
```

2.8. Painterly

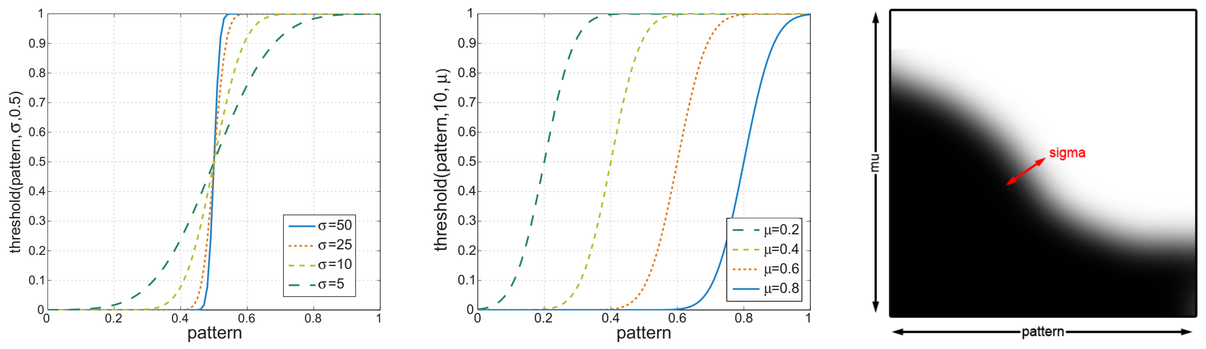


Noise design				
sparse convol.				
impulses/ker..	100	100	100	100
Add widget				
Widget	w1	w2	w3	w4
Seed	120	1235	1233	1234
Gabor				
F_0	0.03100	0.03500	0.20000	0.20000
F_0_w	0.01400	0.00000	0.10000	0.10000
omega_0	0.84540	-1.02460	-1.02460	0.84000
omega_w	0.00000	0.00000	0.00000	0.00000
K_i	0.91700	1.43900	0.38000	0.38100
K_o	0.91700	0.91700	0.91700	0.96800
a	0.03000	0.03000	0.07000	0.07000
isotropic	OFF	OFF	OFF	OFF
kernel radius	65.1006	65.1006	27.9003	27.9003

```
float pattern1 = texture2D(noise_texture0, texCoord).r;
float pattern2 = texture2D(noise_texture1, texCoord).r;
float pattern3 = texture2D(noise_texture2, texCoord).r;
vec3  pattern4 = texture2D(noise_texture3, texCoord).rgb;

vec3  final_color = overlay(color, pattern3, 0.7);

float alpha1 = threshold(pattern1, 25, 0.5);
vec3 stroke1_color = overlayAlpha(final_color, color*0.8, pattern4, 1.2, alpha1);

float alpha2 = threshold(pattern2, 25, 0.5);
final_color = overlayAlpha(stroke1_color, color*0.9, pattern3, 1.2, alpha2);

if(isBackground) final_color = vec3(0.96,0.78,0.5);
```

We can add a bump mapping effect to emphasize the brush fiber layers, similar to the algorithm by Hertzmann [Her02].

3. Threshold Function



(a) Smoothness parameter (σ) (b) Position parameter (μ) (c) Threshold Map

Figure 2: *Threshold functions: (a,b) threshold curves; (c) threshold map*

We use thresholding for various styles. The threshold function is a sigmoid-shaped function. We use a threshold function based on the error function, the integral of a Gaussian function, with parameters μ and σ, which control the position and the smoothness of the threshold respectively (see Fig 2).

```
float threshold(float pattern, float sigma, float mu)
{
  return (erf((sigma * (pattern - mu))) + 1.0) / 2.0;
}
```

Instead of a threshold function, we can also use a *threshold map* (see Fig. 2(c)), an X-toon texture [BTM06] that graphically defines a spatially varying relationship between the threshold function and another variable (for example, shading).

4. Compositing functions

We used the following functions for compositing the noise layers.

Alpha Blending

```
vec3 alphaBlend(vec3 initColor, vec3 colorToBlend, float alpha)
{
  return (1.0-alpha) * initColor + alpha * colorToBlend;
}
```

Overlay

```
vec3 overlay(vec3 initColor, vec3 colorToOverlay, float overlayFactor)
{
  return initColor*overlayFactor*(1.0-(1.0-initColor)*(1.0-colorToOverlay));
}
```

Overlay - Alpha

```
vec3 overlayAlpha(vec3 backColor, vec3 initColor, vec3 colorToOverlay, float overlayFactor, float alpha)
{
  return backColor*(1.0-alpha) + alpha*initColor*overlayFactor*(1.0-(1.0-initColor*alpha)*(1.0-
      colorToOverlay));
}
```

References

[BKTS06] BOUSSEAU A., KAPLAN M., THOLLOT J., SILLION F.: Interactive watercolor rendering with temporal coherence and abstraction. In *Proc. 4th Int. Symposium on Non-Photorealistic Animation and Rendering* (2006), pp. 141–149. 4

[BTM06] BARLA P., THOLLOT J., MARKOSIAN L.: X-toon: An extended toon shader. In *NPAR '2006: international symposium on non-photorealistic animation and rendering* (2006), ACM. 6

[Her02] HERTZMANN A.: Fast paint texture. In *NPAR '2002: international symposium on non-photorealistic animation and rendering* (2002), ACM, p. 91. 5

[Rus04] RUSINKIEWICZ S.: Estimating curvatures and their derivatives on triangle meshes. In *Symposium on 3D Data Processing, Visualization, and Transmission* (2004). 1