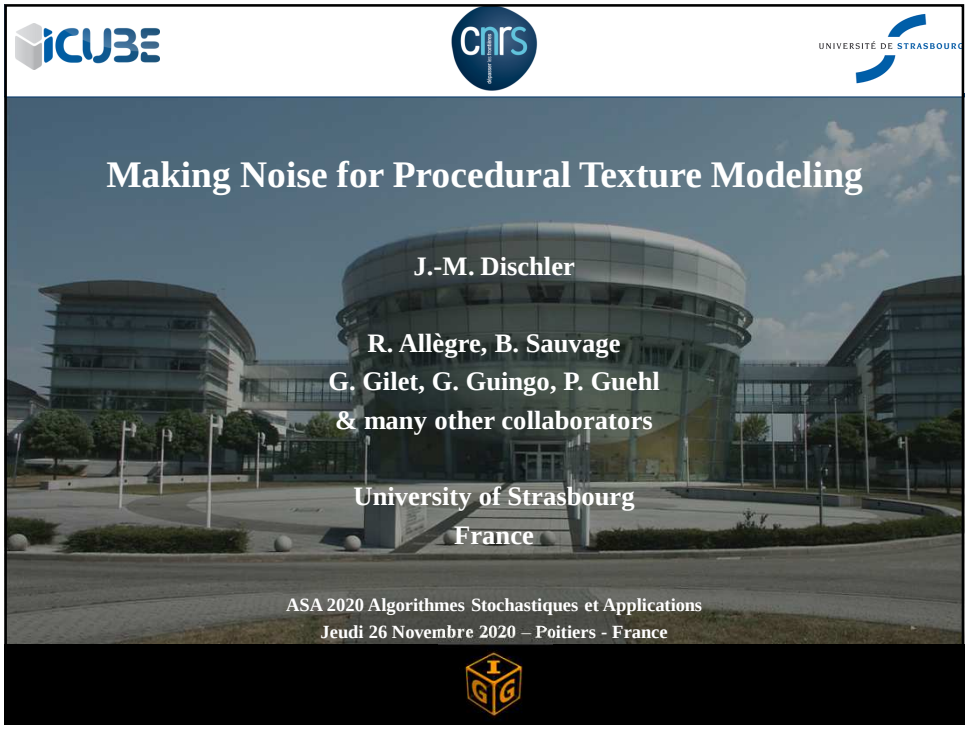
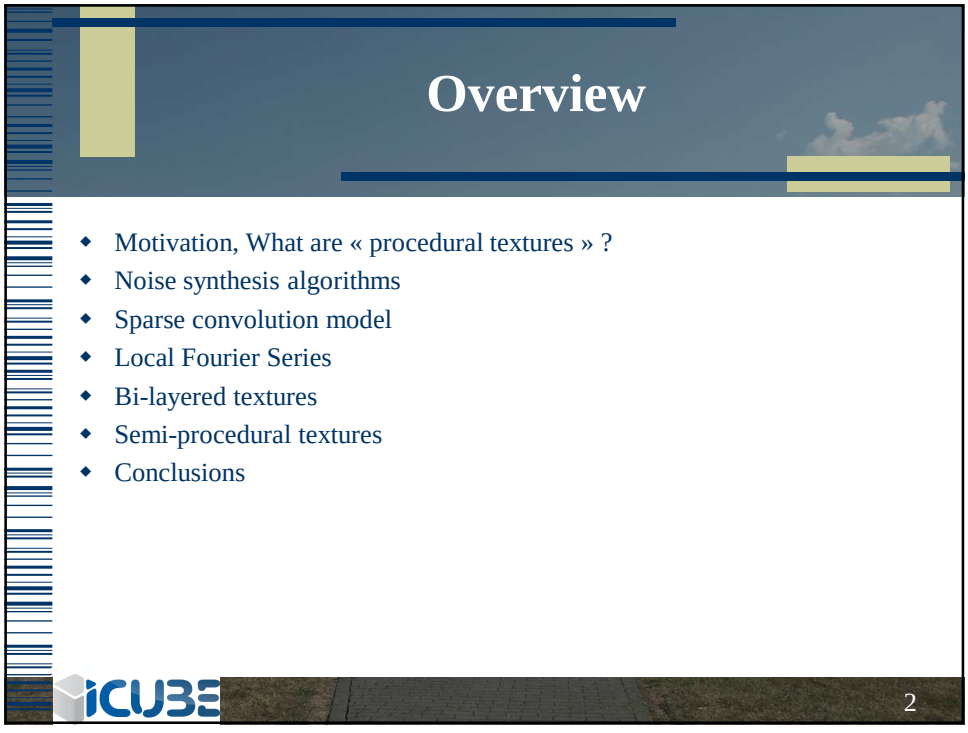

Making Noise for Procedural Texture Modeling
J.-M. Dischler
R. Allègre, B. Sauvage
G. Gilet, G. Guingo, P. Guehl
& many other collaborators
University of Strasbourg
France
ASA 2020 Algorithmes Stochastiques et Applications
Jeudi 26 Novembre 2020 – Poitiers - France

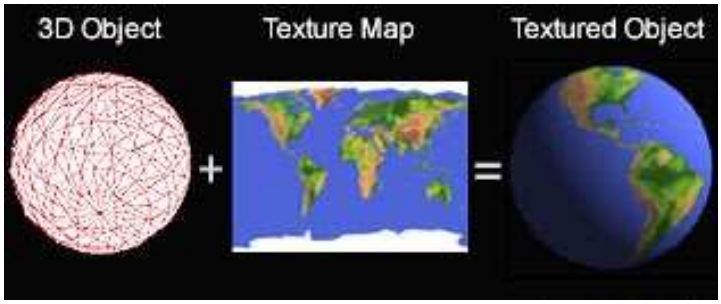
1


Overview
Motivation, What are « procedural textures » ?
Noise synthesis algorithms
Sparse convolution model
Local Fourier Series
Bi-layered textures
Semi-procedural textures
Conclusions
2

2

Historical

Ed Catmull, 1974: texture = image « mapped » on surface using parameterization



3

Why texture synthesis ?

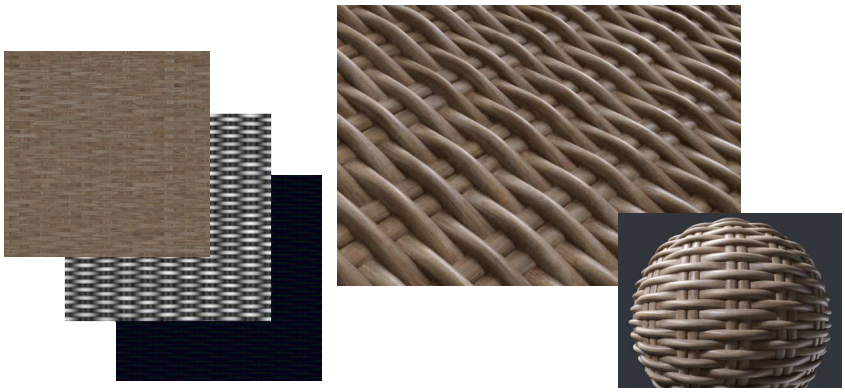
Textures give virtual objects a « material » appearance.

But using images raises many issues…

not realistic

4

Materials: layered images



Source: https://www.textures.com/

5

Why texture synthesis ?

We need **algorithms** to create virtual materials, that have given properties…

more realistic

6

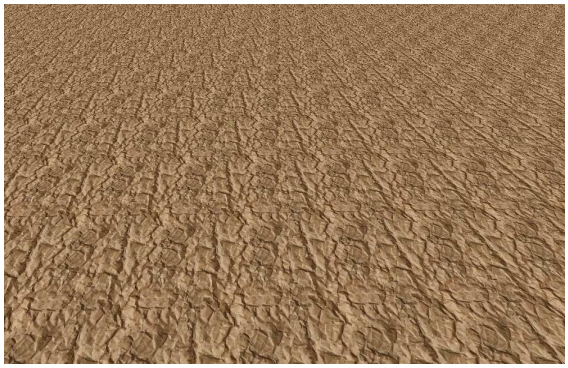
Desired Texture Properties

- *Unbound*
- *Fine details*
- *Fast*
- *Pixel-parallel*
- *Compact*
- *Continuous*
- *Antialising*

7

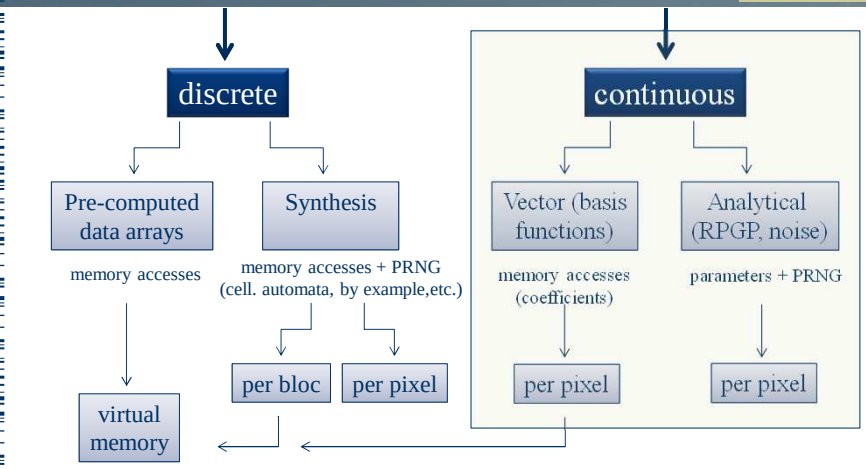
Images have limitations…

- *Not unbound*
- *No fine details*
- *Fast*
- *Pixel-parallel*
- *Not compact*
- *Not continuous*
- *Antialising*



8

Digital texture representations

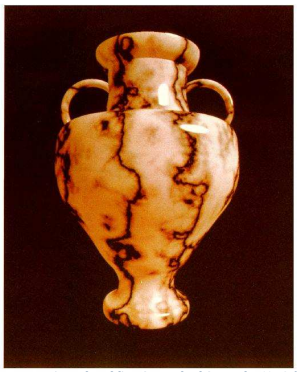


9

Procedural texturing

- **Unbound**
- **Fine details**
- ***Not fast***
- **Pixel-parallel**
- **Compact**
- **Continuous**
- *Antialiasing*

Procedural texture models [Ebert et al.]

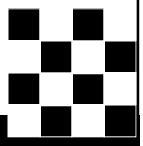


Ebert, David S. and Musgrave, F. Kenton and Peachey, Darwyn and Perlin, Ken and Worley, Steven, **Texturing and Modeling: A Procedural Approach**, 2002, 3rd dition, Morgan Kaufmann Publishers Inc.

10

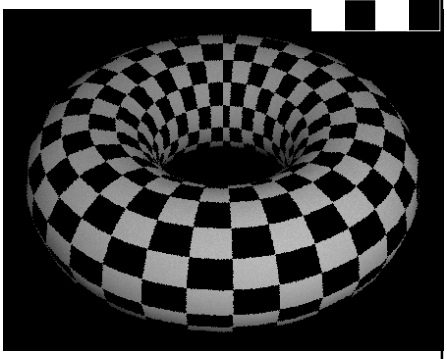
Procedural texturing

Idea: define a texture directly with a program (a function, « **shader** »)



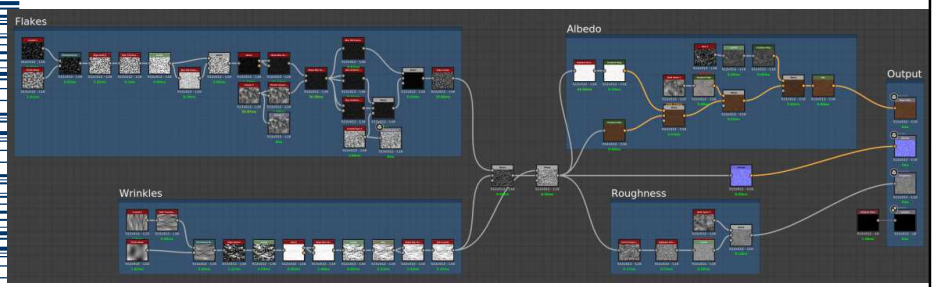
Example : checkerboard

```
Def checker (u,v):
    u=frac(u);
    v=frac(v);
    if (u<0.5 and v<0.5) or
      (u>0.5 and v>0.5):
                return WHITE
    else:
                return BLACK
```



Material Graphs

- **Programming “shaders” is not a trivial task,**
- Shaders are authored by connecting nodes and thereby building a graph;
- Important need: **random functions (stochastic processes)**



https://www.substance3d.com/substance/

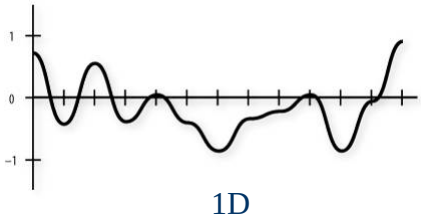
Pseudo Random Number Generation

```
uint seed;

void seeding(uint x, uint y) {
  seed=HASH(x+HASH(y));
}

float next() {
  seed *= uint(1103515245);
  seed=(seed+uint(12345))%(uint(1)<<uint(30));
  seed &= uint(0x3fffffff);
  float res=(float(seed)/float(0x3fffffff))*2.0-1.0;
  return res; // pseudo random value in [-1,1]
}
```

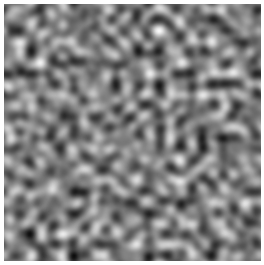

Noise-Functions

Perlin noise, 1985:

Low-pass filtered noise: function $\mathcal{N}$ that returns for a given t a real value in [-1, 1] according to a normal probability distribution.



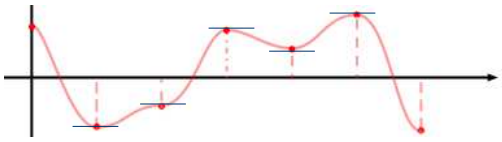
1D



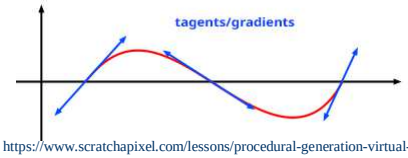
2D

Grid-based noise

Value noise:



Gradient noise:



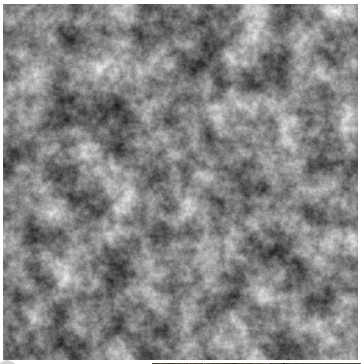
https://www.scratchapixel.com/lessons/procedural-generation-virtual-worlds/procedural-patterns-noise-part-1/simple-pattern-examples

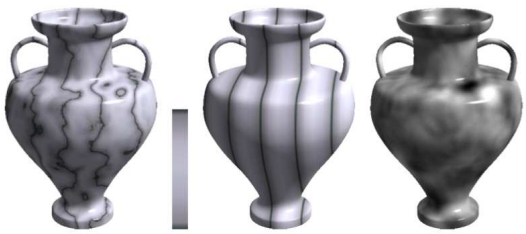
15

Fractal noise

$$turb(t) = \sum_{i=0}^{n} \frac{abs(Bruit(2^i t))}{2^i}$$

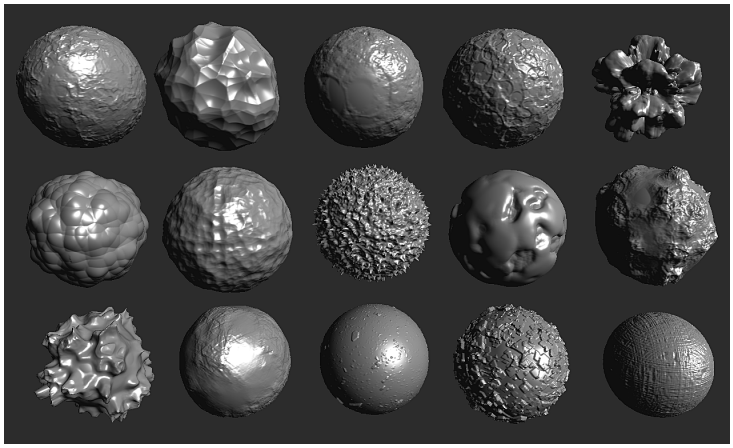
$$Marbre(x,y,z) = \sin^p\left(x + a \cdot turb(b \cdot x, b \cdot y, b \cdot z)\right)$$





16

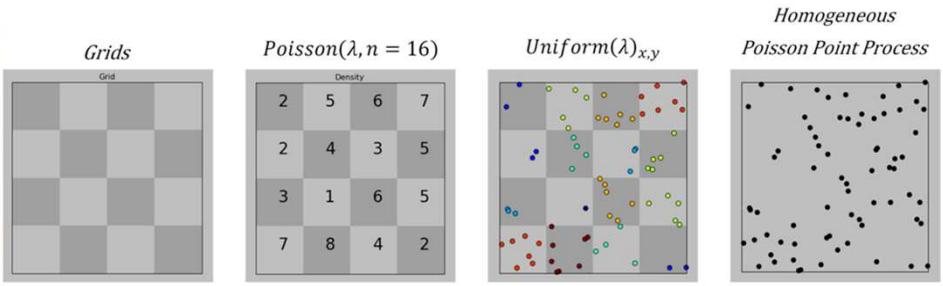
Examples of using different noises for texture modeling



17

Sparse Convolution Noise

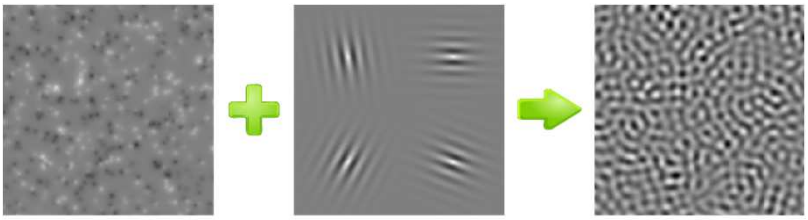
Poisson point process: use an integer lattice and jittering with PRNG to distribute points in space



18

Gabor sparse convolution noise

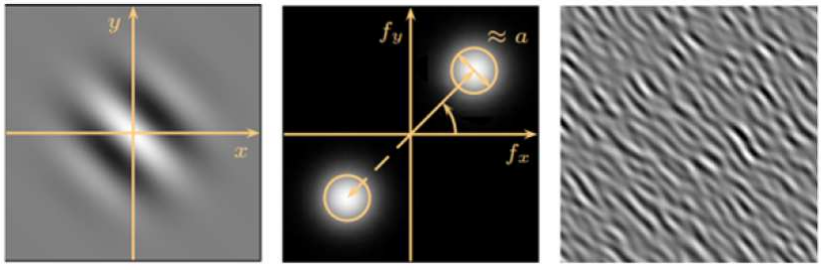
Gabor kernel unifies spatial and spectral properties:



Ares Lagae, Sylvain Lefebvre, George Drettakis, and Philip Dutré. 2009.
Procedural noise using sparse Gabor convolution. ACM Trans. Graph. 28, 3, Article 54 (August 2009)

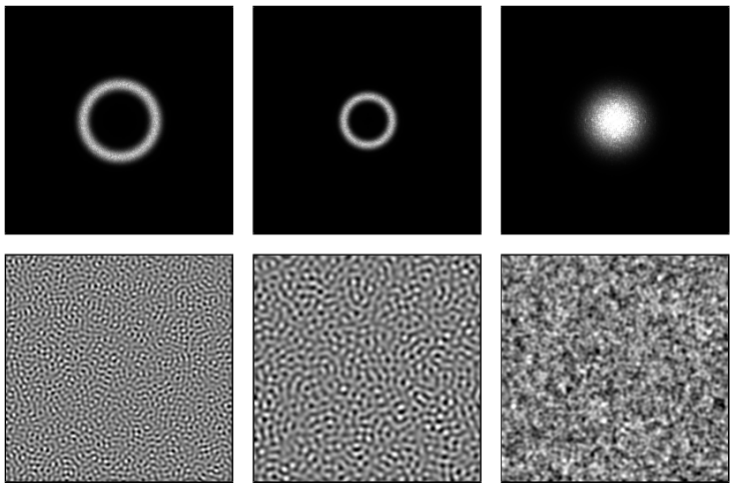
19

Gabor sparse convolution noise



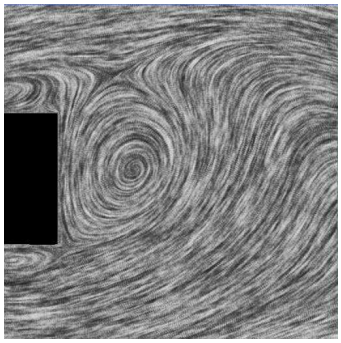
20

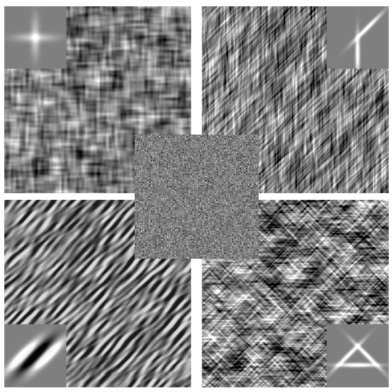
Gabor sparse convolution noise



21

Spot noise (van Wijk, 91)





Arthur Cavalier, Guillaume Gilet, Djamchid Ghazanfarpour: **Local spot noise for procedural surface details synthesis**. Computers and Graphics 85: 92-99 (2019)

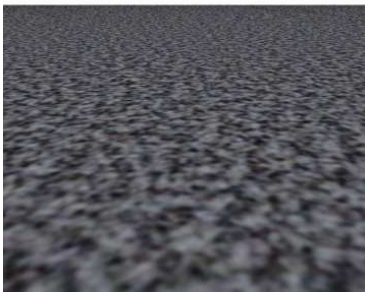
22

Noise « by example »

Noise is hard to use: try to fit parameters of noise using an **example**.

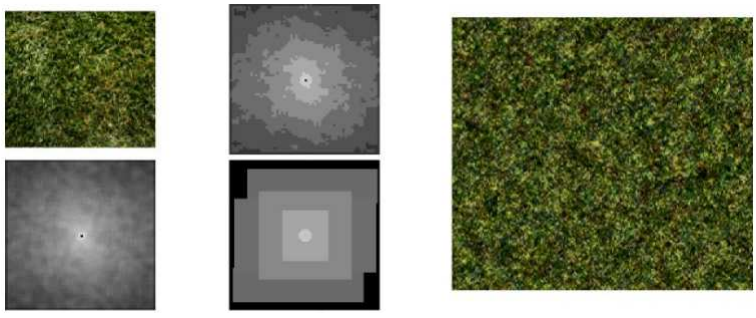
Define a noise that matches: Power Spectral Density & Histogramm of exemplar

Parmeters:
Amplitudes + Frequencies
of noises (PSD)



G Gilet, JM Dischler, L Soler , **Procedural descriptions of anisotropic noisy textures by example**, - Eurographics 2010, Short paper

Noise-Textures by example



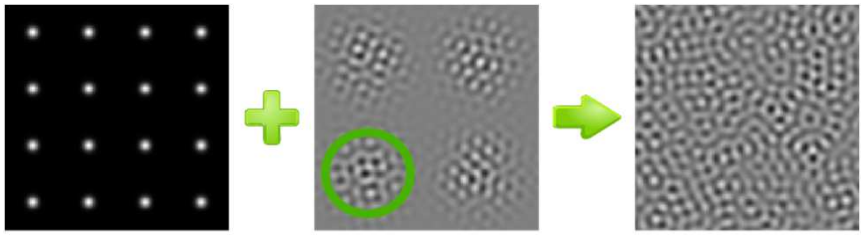
Multiple Kernels Noise for Improved Procedural Texturing
Guillaume Gilet, Jean-Michel Dischler, Djamchid Ghazanfarpour
The Visual Computer, 2012

But only noise, no structure…

25

Local Random-Phase Noise

$$\sum_i w(x - x_i) \times \sum_j (A_{ij} \times cos(f_{ij} \times x + \varphi_{ij}))$$

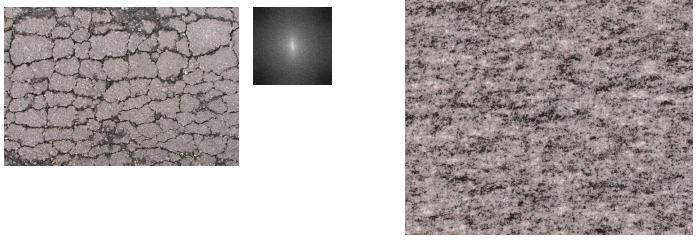


Sum of cosines

26

Local Random-Phase Noise « by example »

Define noise by local Fourier series.

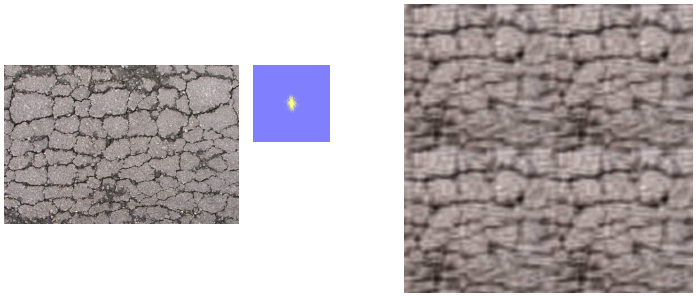


Guillaume Gilet, Basile Sauvage, Kenneth Vanhoey, Jean-Michel Dischler, and Djamchid Ghazanfarpour. 2014. **Local random-phase noise for procedural texturing**. ACM Trans. Graph. 33, 6, (November 2014)

27

Local Random-Phase Noise « by example »

Define noise by local Fourier series.

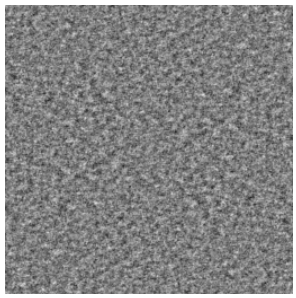


Guillaume Gilet, Basile Sauvage, Kenneth Vanhoey, Jean-Michel Dischler, and Djamchid Ghazanfarpour. 2014. **Local random-phase noise for procedural texturing**. ACM Trans. Graph. 33, 6, (November 2014)

28

Local Random-Phase Noise « by example »

Define noise by local Fourier series.

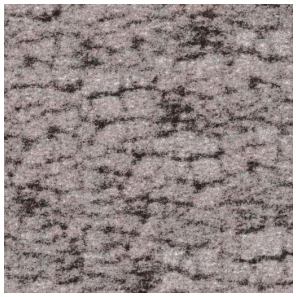


Guillaume Gilet, Basile Sauvage, Kenneth Vanhoey, Jean-Michel Dischler, and Djamchid Ghazanfarpour. 2014. **Local random-phase noise for procedural texturing**. ACM Trans. Graph. 33, 6, (November 2014)

29

Local Random-Phase Noise « by example »

Define noise by local Fourier series.

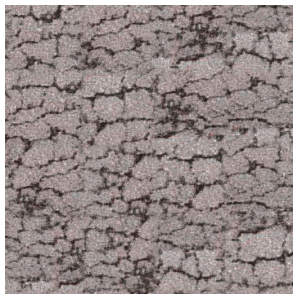


Guillaume Gilet, Basile Sauvage, Kenneth Vanhoey, Jean-Michel Dischler, and Djamchid Ghazanfarpour. 2014. **Local random-phase noise for procedural texturing**. ACM Trans. Graph. 33, 6, (November 2014)

30

Local Random-Phase Noise « by example »

Define noise by local Fourier series.

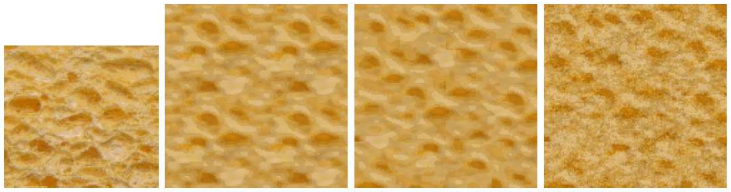


Guillaume Gilet, Basile Sauvage, Kenneth Vanhoey, Jean-Michel Dischler, and Djamchid Ghazanfarpour. 2014. **Local random-phase noise for procedural texturing**. ACM Trans. Graph. 33, 6, (November 2014)

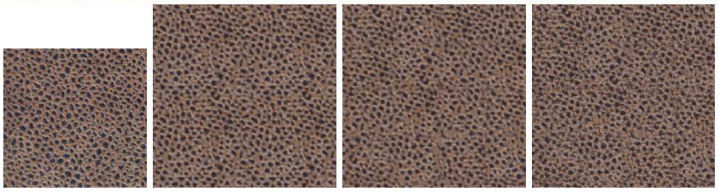
31

Local Random-Phase Noise « by example »

Blending between fast “patch” based texture synthesis (coarse scale) + noise (fine scale).



$J = 50,\ r = 20\%$

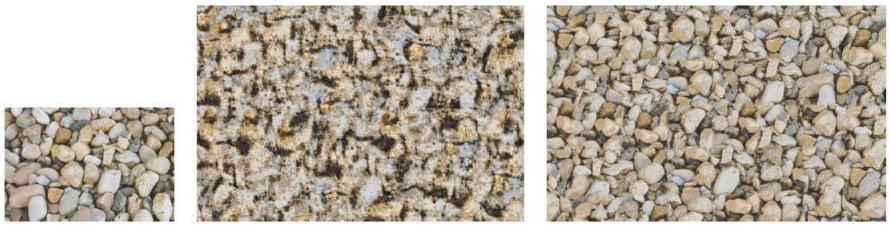


$J = 40,\ r = 35\%$

32

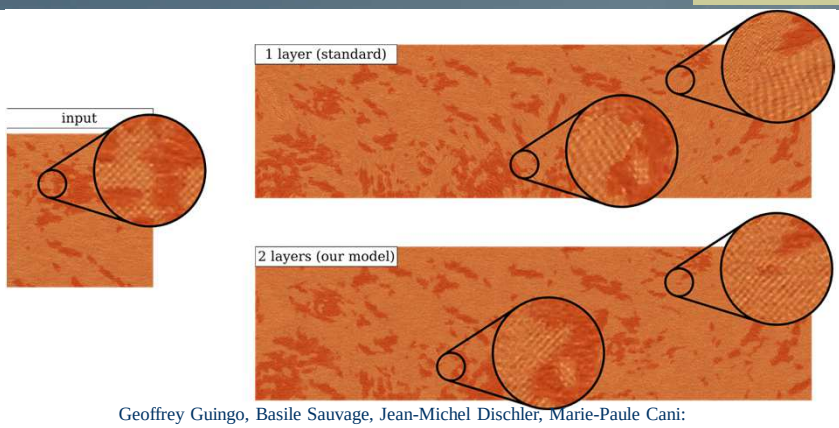
Lack of correlation: « Structure » versus Noise

Un-correlated noise
Too much repetition when fixing too much phases (no real synthesis)



33

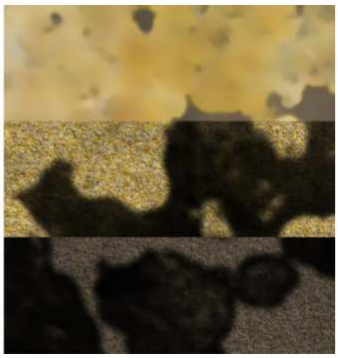
Add visual variety with spatial distorsions



Geoffrey Guingo, Basile Sauvage, Jean-Michel Dischler, Marie-Paule Cani:
Bi-Layer textures: a Model for Synthesis and Deformation of Composite Textures. Comput. Graph. Forum 36(4): 111-122 (2017)

34

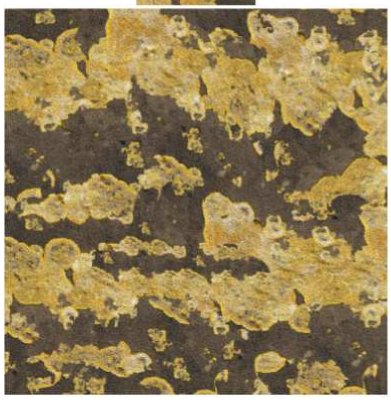
Bi-layered Texture synthesis



Geoffrey Guingo, Basile Sauvage, Jean-Michel Dischler, Marie-Paule Cani:
Bi-Layer textures: a Model for Synthesis and Deformation of Composite Textures. Comput. Graph. Forum 36(4): 111-122 (2017)

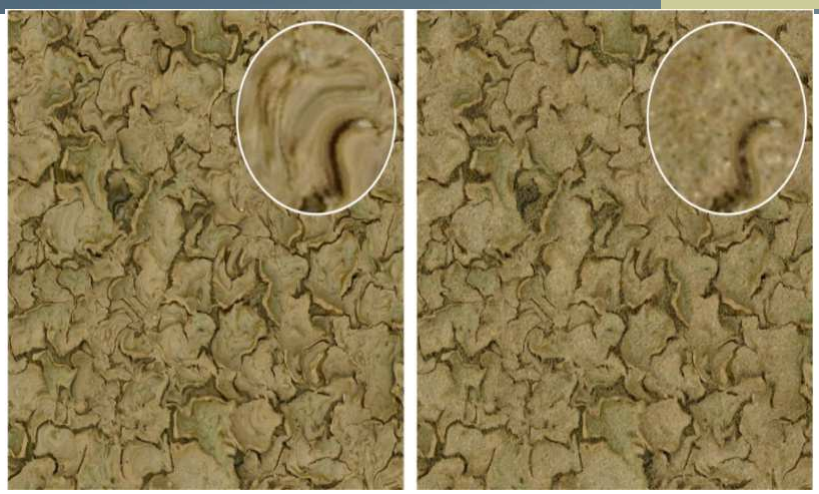
35

Bi-layered Texture synthesis



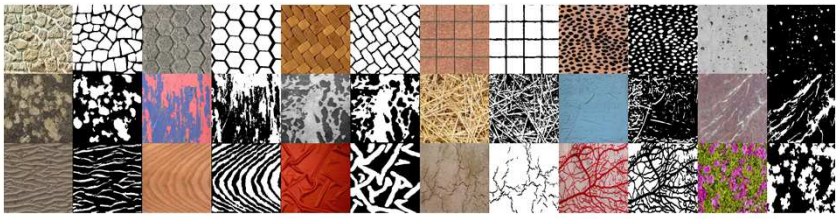
36

Bi-layered Texture synthesis



37

Structure versus Noise



Many natural textures embed *spatial stochastic structures* such as cells, cracks, grains, scratches, spots, stains or waves.

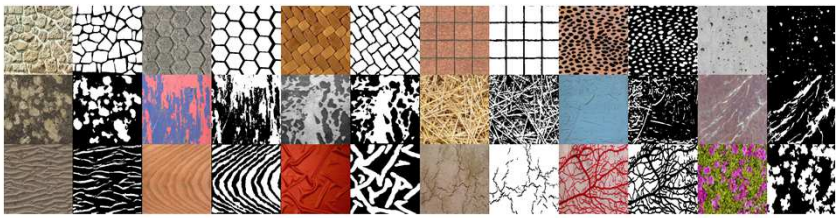
They can be characterized by *binary images*

Three characteristics:

1. Distribution
2. Mutual interaction
3. Local visual shape

38

Structure versus Noise



$$Structure(\vec{x}) = P * (W \ F)(\vec{x})$$

(1) (2) (3)

Three characteristics:

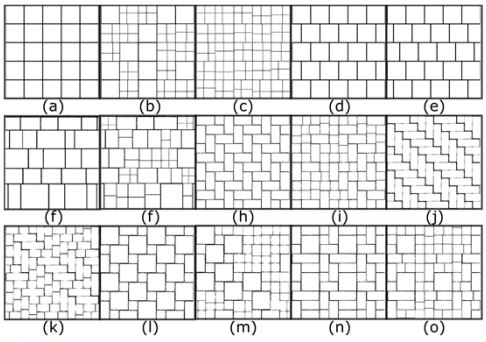
1. Distribution
2. Mutual interaction
3. Local visual shape

39

Non-Uniform Point Distributions

POINT PROCESS

$$P(\vec{x}) = \sum_i \delta(\vec{x} - \vec{x}_i)$$



Poisson point process

approximations : square cells with constant area

Cox point process

random tessellations, cells consist of rectangles Ri of varying sizes

Gibbs point process

approximation : (iterative) Lloyd algorithm

Near-regular / regular

Jittering amplitude controls randomness
low value can model near-regular distributions

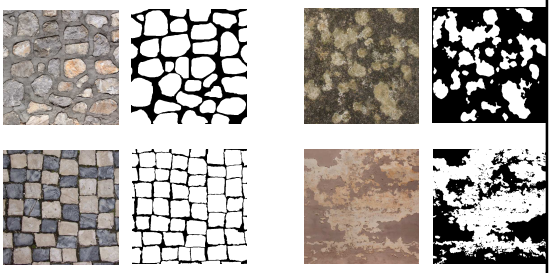
40

Window Function

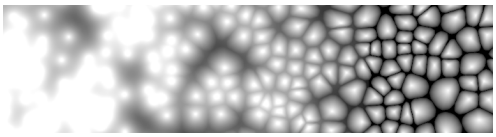
$$w(\mathbf{x}) = \sum_{j=1}^{n_w} \frac{\omega_j}{\mathrm{N}_j} w_j(\mathbf{x})$$

$$w_j = \lfloor e^{-\sigma_j D_j(\mathbf{x}-\vec{x_i})} \rfloor_{d_j}$$

weighted sum of collections of basis windows with Gaussian profile



overlapping and *non-overlapping*



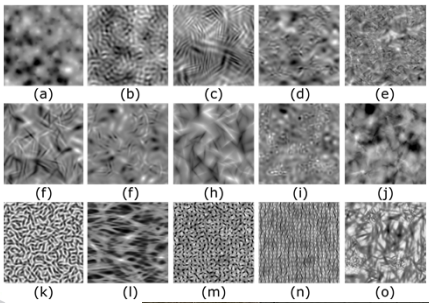
41

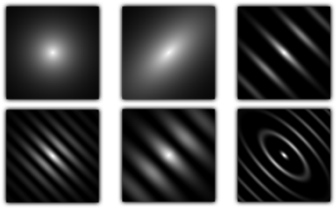
Feature Function

$$f(\mathbf{x}) = \sum_{j=1}^{J} \omega_j(\mathbf{x}) \tilde{G}_j(\mathbf{x}),$$

$$\tilde{G}_j(\mathbf{x}) = A_j \left(e^{-\sigma_{j|i} \left\| \Xi(\vec{x} - \vec{\mu_{j|i}}) \right\|_f} \right) \left(.5 + .5 \cos \left(\phi \left\| \Xi(\vec{x} - \vec{\mu_{j|i}}) \right\|_f \right) \right)^{\tau}.$$

Anisotropy, *thickness* and *curliness* allow more efficient modeling of anisotropic features like grains, stripes and folds.





42

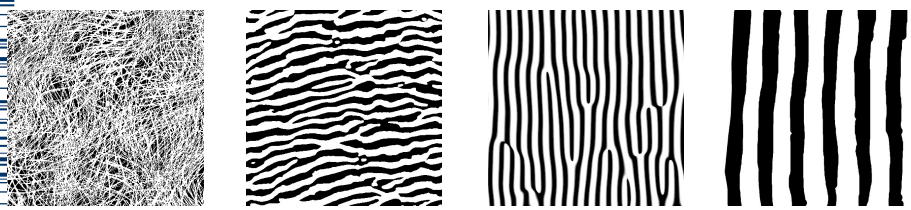
Correlating Window and Feature Function

No correlation

adding little correlations helps spatial organization

adding more correlations

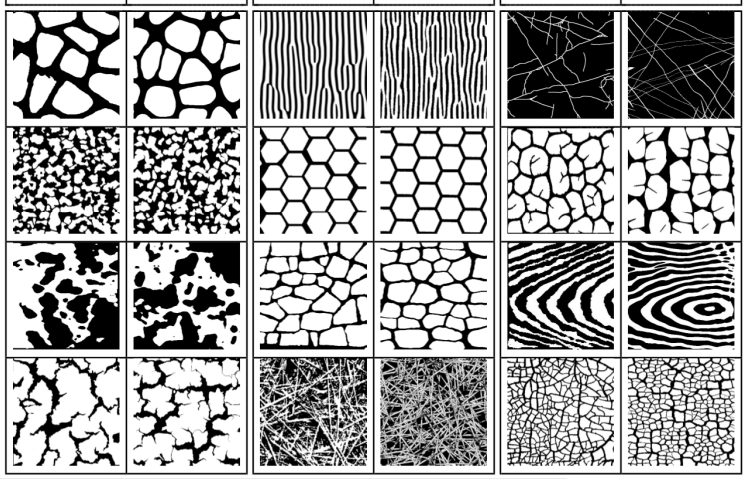
strong correlation



43

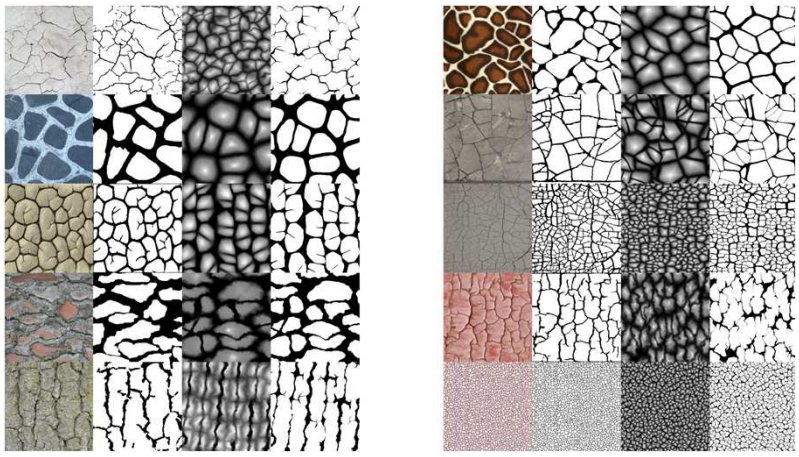
Evaluation of visual « span »

Input binary structure	Procedural structure	Input binary structure	Procedural structure	Input binary structure	Procedural structure



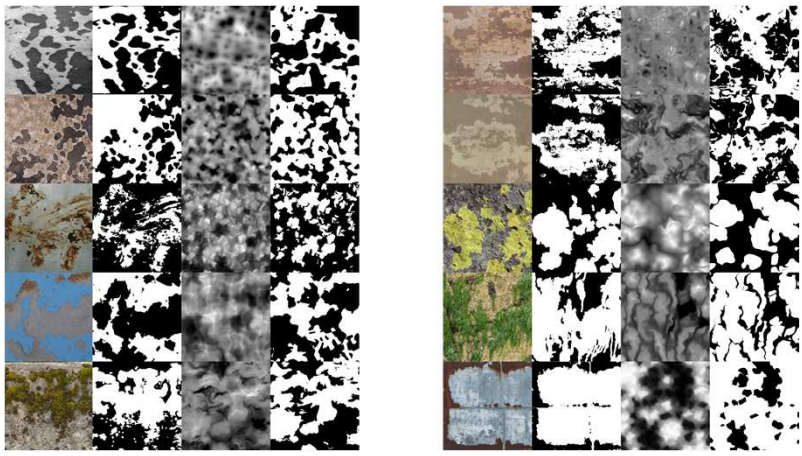
44

Structure classes: cells, grains, lines, waves, stains, piles, dots, tilings



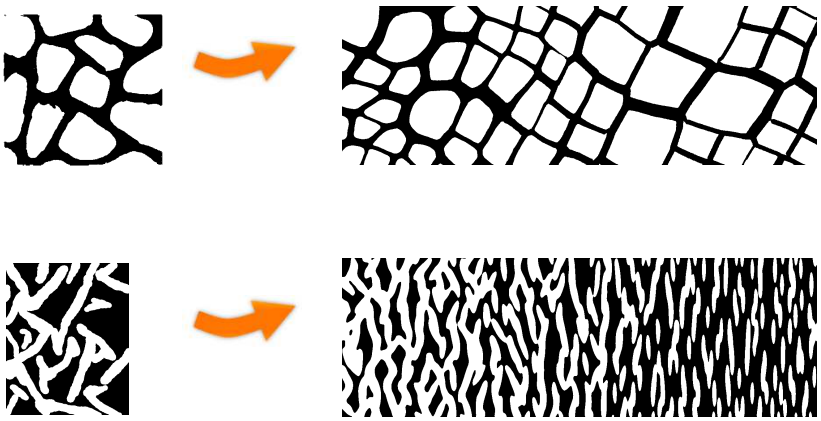
45

Structure classes: cells, grains, lines, waves, stains, piles, dots, tilings



46

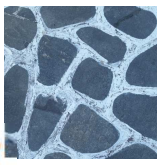
Edition + Control



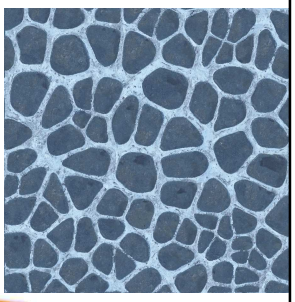
47

Semi-procedural texture

Input exemplar

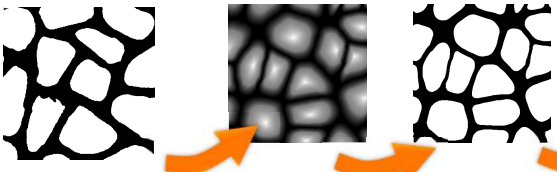


Synthesized output
(semi-procedural texture)



Procedural structure *(PPTBF)*

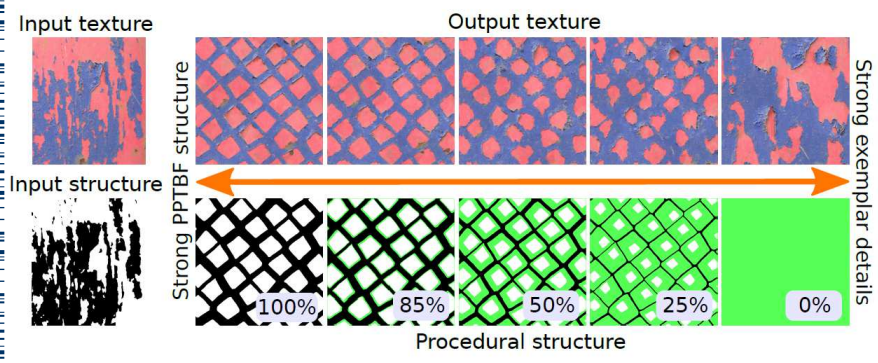
Segmented input (user provided)



Color transfer

48

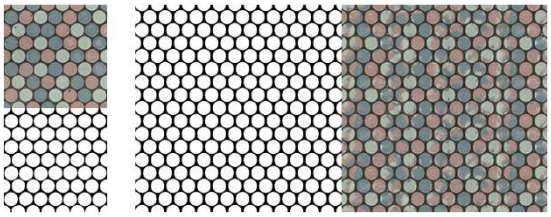
Procedural Structure versus Exemplar



49

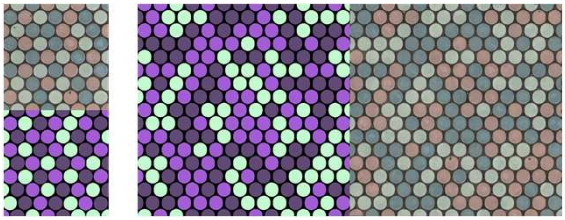
Label-maps

NO label map



mix of colors

WITH label map



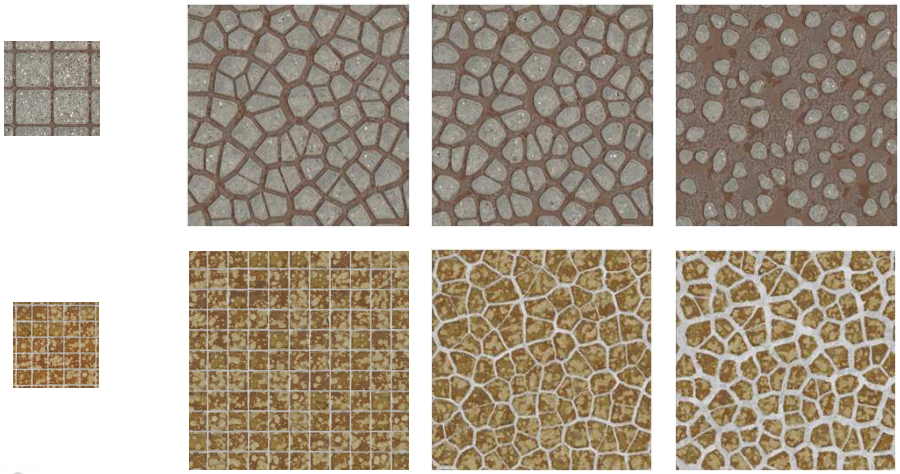
no more mix

50

Synthesis Results

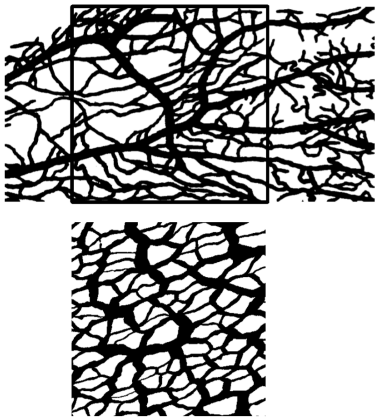
51

Modify structure



52

Limitations



Conclusion

Contributions

- Local Fourier Series Noise
- Procedural stochastic structure model
- Semi-procedural texture synthesis
- Database of segmented exemplars
- Database of synthesis results
- Source code available

Future work

- Explore more “non-Gaussian” stochastic processes
- Using machine learning for parameter estimation