Lectures on Stochastic Methods for Image Analysis

Agnès Desolneux

CNRS, CMLA, ENS Cachan

4th Stochastic Geometry Days, Poitiers, 24-26 August 2015





◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 三臣 - のへ(?)

LAST PART : TEXTURE IMAGES AND RANDOM FIELDS

Special thanks to Bruno Galerne for giving me his slides !

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 臣 のへで

Outline

Texture synthesis

Patch-based synthesis algorithms

Discrete Fourier transform of digital images

Random phase noise (RPN)

Asymptotic discrete spot noise (ADSN)

RPN and ADSN as texture synthesis algorithms

Numerical experiments

Texton for RPN and ADSN texture analysis and synthesis

(ロ) (同) (三) (三) (三) (○) (○)

What is a texture?

A minimal definition of a **texture** image is an "image containing repeated patterns" [Wei et al., 2009].

The family of patterns reflects a certain amount of randomness, depending on the nature of the texture.

Two main subclasses :

▶ The *micro-textures*.



> The *macro-textures*, constitued of small but discernible objects.







Textures and scale of observation

Depending on the viewing distance, the same objects can be perceived either as

- a micro-texture,
- a macro-texture,
- a collection of individual objects.



Micro-texture

Macro-texture

▲□▶▲□▶▲□▶▲□▶ □ のQ@

Texture synthesis

Texture Synthesis : Given an input texture image, produce an output texture image being both visually similar to and pixel-wise different from the input texture.



The output image should ideally be perceived as another part of the same large piece of homogeneous material the input texture is taken from.

ヘロア 人間 アメヨア 人間 アーヨ

Texture synthesis algorithms

Two main kinds of algorithm :

Texture synthesis algorithms

Two main kinds of algorithm :

- 1. Neighborhood-based synthesis algorithms (or "copy-paste" algorithms) : Algorithm :
 - Compute sequentially an output texture such that each patch of the output corresponds to a patch of the input texture.
 - Many variations have been proposed : scanning orders, grow pixel by pixel or patch by patch, multiscale synthesis, optimization procedure,...

(ロ) (同) (三) (三) (三) (○) (○)

Properties :

- + Synthesize well macro-textures
- Can have some speed and stability issue, hard to set parameter...

Texture synthesis algorithms

Two main kinds of algorithm :

- 1. Neighborhood-based synthesis algorithms (or "copy-paste" algorithms) : Algorithm :
 - Compute sequentially an output texture such that each patch of the output corresponds to a patch of the input texture.
 - Many variations have been proposed : scanning orders, grow pixel by pixel or patch by patch, multiscale synthesis, optimization procedure,...

Properties :

- + Synthesize well macro-textures
- Can have some speed and stability issue, hard to set parameter...
- 2. Texture synthesis using statistical constraints :

Algorithm :

- 2.1 Extract some meaningful "statistics" from the input image (e.g. distribution of colors, of Fourier coefficients, of wavelet coefficients,...).
- 2.2 Compute a "random" output image having the same statistics : start from a white noise and alternatively impose the "statistics" of the input.

Properties :

- + Perceptually stable
- Generally not good enough for macro-textures
- + Allows mathematical computations !

Important main paper : "Texture synthesis by non-parametric sampling" by Efros and Leung, 1999.

▲□▶▲□▶▲□▶▲□▶ □ のQ@

- First paper with patch-based algorithm
- Concept of redundancy of patches in images
- Strong influence for many applications
 - Inpainting
 - Image editing
 - Denoising

Back to Shannon's work on English text

Shannon's approach for text synthesis : a Markov chain model.

- 1. Draw an initial *n*-gram from the marginal distribution on *n*-grams $P^{(n)}$ (learnt from a large database of texts).
- 2. Add characters sequentially using the conditional distribution on *n*-grams given the first n 1 characters :

$$P^{(n)}(a_n | a_1 \dots a_{n-1}) = \frac{P^{(n)}(a_1 \dots a_{n-1}a_n)}{\sum_b P^{(n)}(a_1 \dots a_{n-1}b)}$$

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

Samples from the *n*-gram distributions

Random characters (Sample from P⁽⁰⁾): XFOML RXKHRJFFJUJ ZLPWCFWKCYJ FFJEYVKCQSGXYD QPAAMKBZAACIBZLHJQD

Sample from P⁽¹⁾

OCRO HLI RGWR NMIELWIS EU LL NBBESEBYA TH EEI ALHENHTTPA OO BTTV

▶ Sample from P⁽²⁾

ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE TUCOOWE FUSO TIZIN ANDY TOBE SEACE CTISBE

Sample from P⁽³⁾

IN NO IST LAY WHEY CRATICT FROURE BERS GROCID PONDENOME OF DEMONSTURES OF THE REPTAGIN IS REGOACTIONA OF CRE

Sample from P⁽⁴⁾

THE GENERATED JOB PROVIDUAL BETTER TRAND THE DISPLAYED CODE ABOVERY UPONDULTS WELL THE CODERST IN THESTICAL IT TO HOCK BOTHE

Shannon's approach for Texture Synthesis?

Differences between texture images and texts for applying Shannon's approach :

- No natural order in a 2D pixel grid.
- Pixel similarity : two close pixel values can be exchanged without degrading the texture while it is not possible for udys (=text +/- 1).

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

Computation of conditional probabilities becomes uneasy.

But Markov modeling is still OK !

Assuming Markov property, one wants to estimate $Prob(I(p)|I(\mathcal{N}(p)))$.

Just search in the input original image all similar neighborhoods - that's an estimate of the probability distribution of I(p) knowing its neighborhood.

To sample from this probability distribution, just pick one match at random.

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

Efros-Leung algorithm

Let I_0 be an original input image.

Input : size of the ouput image *I*, size of the neighborhood, precision parameter ε (= 0.1 in general).

- 1. Initialize with a random patch from I_0
- 2. While the output I is not filled
 - 2.1 Pick an unknown pixel p in I with maximal known neighbor pixels, denoted $\mathcal{N}(p)$.
 - 2.2 Compute the set of q in I_0 such that

 $d(I_0(\mathcal{N}(q)), I(\mathcal{N}(p))) \le (1+\varepsilon) \min_{x} d(I_0(\mathcal{N}(x)), I(\mathcal{N}(p)))$

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

2.3 Randomly pick one of this *q* and set $I(p) = I_0(q)$.

Non-parametric Sampling



Influence of the size of the neighborhood



(日)

Examples of results



Homage to Shannon



Failure examples : « Garbage » or « Verbatim copy »



▲□▶▲圖▶▲≣▶▲≣▶ ≣ の�?

Influence of the size of the neighborhood



◆□ > ◆□ > ◆豆 > ◆豆 > ̄豆 − のへぐ

Heeger-Bergen algorithm [Heeger and Bergen, 1995]

Statistical constraints :

- Histogram of colors.
- Histogram of wavelet coefficients at each scale.

Algorithm : Alternating projections into the constraints starting from a white noise image.



Texture synthesis by phase randomization

What about the Random Phase Noise (RPN) and Asymptotic Discrete Spot Noise (ADSN) presented today?

[Galerne, Gousseau and Morel, 2011 (a)] [Galerne, Gousseau and Morel, 2011 (b)]

◆□▶ ◆□▶ ▲□▶ ▲□▶ □ のQ@

- It belongs to the first category : texture synthesis by statistical constraints.
- Here the "statistics" are the moduli of the Fourier coefficients.

Texture synthesis by phase randomization

Successful examples with micro-textures



Texture synthesis by phase randomization

► Failure examples with macro-textures



Framework

- ► We work with discrete digital images $u \in \mathbb{R}^{M \times N}$ indexed on the set $\Omega = \{0, ..., M 1\} \times \{0, ..., N 1\}.$
- Each image is extended by periodicity :

 $u(k, l) = u(k \mod M, l \mod N)$ for all $(k, l) \in \mathbb{Z}^2$.

Consequence : Translation of an image :





- Image domain : $\Omega = \{0, \dots, M-1\} \times \{0, \dots, N-1\}$
- Fourier domain $\hat{\Omega}$: the frequency 0 is placed at the center :

$$\hat{\Omega} = \left\{-\frac{M}{2}, \dots, \frac{M}{2} - 1\right\} \times \left\{-\frac{N}{2}, \dots, \frac{N}{2} - 1\right\}.$$

Definition :

• The discrete Fourier transform (DFT) of u is the complex-valued image \hat{u} defined by :

$$\hat{u}(s,t) = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} u(k,l) e^{-\frac{2ik\pi}{M}} e^{-\frac{2ilt\pi}{N}}, \quad (s,t) \in \hat{\Omega}.$$

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ - 三 - のへぐ

- Image domain : $\Omega = \{0, ..., M 1\} \times \{0, ..., N 1\}$
- Fourier domain $\hat{\Omega}$: the frequency 0 is placed at the center :

$$\hat{\Omega} = \left\{-\frac{M}{2}, \dots, \frac{M}{2} - 1\right\} \times \left\{-\frac{N}{2}, \dots, \frac{N}{2} - 1\right\}.$$

Definition :

• The discrete Fourier transform (DFT) of u is the complex-valued image \hat{u} defined by :

$$\hat{u}(s,t) = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} u(k,l) e^{-\frac{2ik\pi}{M}} e^{-\frac{2ilt\pi}{N}}, \quad (s,t) \in \hat{\Omega}.$$

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

- ▶ $|\hat{u}|$: Fourier modulus of *u*.
- $\arg(\hat{u})$: Fourier phase of u.

- Image domain : $\Omega = \{0, ..., M 1\} \times \{0, ..., N 1\}$
- Fourier domain $\hat{\Omega}$: the frequency 0 is placed at the center :

$$\hat{\Omega} = \left\{-\frac{M}{2}, \dots, \frac{M}{2} - 1\right\} \times \left\{-\frac{N}{2}, \dots, \frac{N}{2} - 1\right\}.$$

Definition :

► The discrete Fourier transform (DFT) of *u* is the complex-valued image \hat{u} defined by :

$$\hat{u}(s,t) = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} u(k,l) e^{-\frac{2ik\pi}{M}} e^{-\frac{2ilt\pi}{N}}, \quad (s,t) \in \hat{\Omega}.$$

- $|\hat{u}|$: Fourier modulus of u.
- $\arg(\hat{u})$: Fourier phase of u.

Symmetry property :

► Since *u* is real-valued, $\hat{u}(-s, -t) = \hat{u}(s, t)$. ⇒ the modulus $|\hat{u}|$ is even and the phase $\arg(\hat{u})$ is odd.

Symmetry property :

- $|\hat{u}|$: Fourier modulus of *u* is even.
- $\arg(\hat{u})$: **Fourier phase** of *u* is odd.

Visualization of the DFT :



Image u



Modulus $|\hat{u}|$



Phase $\arg(\hat{u})$

◆□▶ ◆□▶ ▲□▶ ▲□▶ □ のQ@

Symmetry property :

- $|\hat{u}|$: Fourier modulus of *u* is even.
- $\arg(\hat{u})$: **Fourier phase** of *u* is odd.

Visualization of the DFT :







Image u

Modulus $|\hat{u}|$

Phase $\arg(\hat{u})$

Computation :

- ► The Fast Fourier Transform algorithm computes \hat{u} in $\mathcal{O}(MN \log(MN))$ operations.
- Efficient FFT implementation : FFTW library, a c/c++ library (used in Matlab).

FFTW = Fastest Fourier Transform in the West

Exchanging the modulus and the phase of two images : [Oppenheim and Lim, 1981]

Image 1



Image 2

Modulus of 1 & phase of 2

Modulus of 2 & phase of 1

▲□▶▲□▶▲□▶▲□▶ □ のQ@

Exchanging the modulus and the phase of two images : [Oppenheim and Lim, 1981]



< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

Geometric contours are mostly contained in the phase.

Exchanging the modulus and the phase of two images : [Oppenheim and Lim, 1981]

Image 1

Modulus of 1 & phase of 2



Modulus of 2 & phase of 1

Textures are mostly contained in the modulus.

Exchanging the modulus and the phase of two images : [Oppenheim and Lim, 1981]



- Geometric contours are mostly contained in the phase.
- Textures are mostly contained in the modulus.

Random phase textures

- We call random phase texture any image that is perceptually invariant to phase randomization.
- Phase randomization = replace the Fourier phase by a random phase.
- Definition : A random field $\theta : \hat{\Omega} \to \mathbb{R}$ is a *random phase* if
 - 1. Symmetry : θ is odd :

$$\forall (s,t) \in \hat{\Omega}, \theta(-s,-t) = -\theta(s,t).$$

- 2. Distribution : Each component $\theta(s, t)$ is
 - uniform over the interval $] \pi, \pi]$ if $(s, t) \notin \{(0, 0), (\frac{M}{2}, 0), (0, \frac{N}{2}), (\frac{M}{2}, \frac{N}{2})\},\$
 - uniform over the set $\{0, \pi\}$ otherwise.
- Independence : For each subset S ⊂ Ω̂ that does not contain distinct symmetric points, the r.v. {θ(s, t)|(s, t) ∈ S} are independent.
- Property : The Fourier phase of a Gaussian white noise X is a random phase.
- (Lazy) simulation : In Matlab, theta = angle(fft2(randn(M,N))).
- Random phase textures constitute a "limited" subclass of the set of textures.

Random Phase Noise (RPN)

- Texture synthesis algorithm : random phase noise (RPN) : [van Wijk, 1991]
- 1. Compute the DFT \hat{h} of the input *h*.
- 2. Compute a random phase θ using a pseudo-random number generator.

3. Set
$$\hat{Z} = \left| \hat{h} \right| e^{i\theta}$$
 (or $\hat{Z} = \hat{h}e^{i\theta}$).

4. Return Z the inverse DFT of \hat{Z} .



(ロ) (同) (三) (三) (三) (○) (○)

Discrete spot noise [van Wijk, 1991]

- Let *h* be a discrete image called *spot*.
- Let (X_k) be a sequence of random translation vectors which are i.d.d. and uniformly distributed over Ω .
- The discrete spot noise of order n associated with h is the random image

$$f_n(x) = \sum_{k=1}^n h(x - X_k).$$

(translations with periodic boundary conditions)



Limit of the DSN model?



- For texture synthesis we are more particularly interested in the limit of the DSN: the asymptotic discrete spot noise (ADSN).
- ▶ The *DSN* of order $n, f_n(x) = \sum_k h(x X_k)$, is the sum of the *n* i.i.d. random images $h(\cdot X_k)$.
- Central limit theorem for random vectors :

The sequence of random images $\left(\frac{f_n - n\mathbb{E}(h(\cdot - X_1))}{\sqrt{n}}\right)_{n \in \mathbb{N}^*}$ converges in distribution towards the **Gaussian random vector** $Y = (Y(x))_{x \in \Omega}$ with zero mean and covariance $Cov(h(\cdot - X_1))$.

◆□▶ ◆□▶ ◆∃▶ ◆∃▶ = のへで

Asymptotic discrete spot noise (ADSN)

Expectation of the random translations :

$$\mathbb{E}(h(x - X_1)) = \sum_{y \in \Omega} h(x - y) \mathbb{P}(X_1 = y)$$
$$= \sum_{y \in \Omega} h(x - y) \frac{1}{MN}$$
$$= \frac{1}{MN} \sum_{z \in \Omega} h(z)$$
$$= \text{mean of } h.$$

• $\mathbb{E}(h(x - X_1)) = m$, where *m* is the mean of *h*.

Asymptotic discrete spot noise (ADSN)

Covariance of the random translations : Let $x, y \in \Omega$,

$$Cov(h(x - X_1), h(y - X_1)) = \mathbb{E}((h(x - X_1) - m)(h(y - X_1) - m))$$

= $\sum_{z \in \Omega} (h(x - z) - m)(h(y - z) - m)\mathbb{P}(X_1 = z)$
= $\frac{1}{MN} \sum_{z \in \Omega} (h(x - z) - m)(h(y - z) - m)$
= $C_h(x, y).$

• $Cov(h(x - X_1), h(y - X_1)) = C_h(x, y)$ where C_h is the **autocorrelation** of h:

$$C_h(x,y) = \frac{1}{MN} \sum_{t \in \Omega} \left(h(x-t) - m \right) \left(h(y-t) - m \right), \quad (x,y) \in \Omega.$$

Asymptotic discrete spot noise (ADSN)

For texture synthesis we are more particularly interested in the limit of the DSN: the asymptotic discrete spot noise (ADSN).

Expectation and covariance of the random translations :

- $\mathbb{E}(h(x X_1)) = m$, where *m* is the arithmetic mean of *h*.
- $Cov(h(x X_1), h(y X_1)) = C_h(x, y)$ where C_h is the autocorrelation of h:

$$C_h(x,y) = \frac{1}{MN} \sum_{t \in \Omega} (h(x-t) - m) (h(y-t) - m) = C_h(0, y - x).$$

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

Definition of ADSN :

• The ADSN associated with h is the Gaussian vector $\mathcal{N}(0, C_h)$.

Simulation of the ADSN

Definition of *ADSN* : the *ADSN* associated with *h* is the Gaussian vector $\mathcal{N}(0, C_h)$.



Gaussian white noise : pixels are independent and have Gaussian distribution



Gaussian vector : pixels have Gaussian distribution and are correlated

◆□▶ ◆□▶ ◆ □▶ ◆ □▶ ─ □ ─ つへぐ

Convolution product :
$$(f * g)(x) = \sum_{y \in \Omega} f(x - y)g(y), x \in \Omega.$$

Simulation of the ADSN :

► Let $h \in \mathbb{R}^{M \times N}$ be a an image, *m* be the mean of *h* and *X* be a Gaussian white noise image.

► The random image $\frac{1}{\sqrt{MN}} (h - m) * X$ is the *ADSN* associated with *h*.



Proof of
$$Y = \frac{1}{\sqrt{MN}} (h - m) * X \simeq \mathcal{N}(0, C_h).$$

▶ *Y* is obtained from *X* by applying a linear mapping. Since *X* is a Gaussian vector, *Y* is also a Gaussian vector.

Proof of
$$Y = \frac{1}{\sqrt{MN}} (h - m) * X \simeq \mathcal{N}(0, C_h).$$

► *Y* is obtained from *X* by applying a linear mapping. Since *X* is a Gaussian vector, *Y* is also a Gaussian vector.

• One just needs to show that $\mathbb{E}(Y(x)) = 0$ and $Cov(Y(x), Y(y)) = C_h(x, y)$.

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

Proof of
$$Y = \frac{1}{\sqrt{MN}} (h - m) * X \simeq \mathcal{N}(0, C_h).$$

- ► *Y* is obtained from *X* by applying a linear mapping. Since *X* is a Gaussian vector, *Y* is also a Gaussian vector.
- One just needs to show that $\mathbb{E}(Y(x)) = 0$ and $Cov(Y(x), Y(y)) = C_h(x, y)$.

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

► By linearity,
$$\mathbb{E}(Y(x)) = \frac{1}{\sqrt{MN}} (h - m) * \mathbb{E}(X)(x) = 0.$$

Proof of
$$Y = \frac{1}{\sqrt{MN}} (h - m) * X \simeq \mathcal{N}(0, C_h).$$

- ▶ *Y* is obtained from *X* by applying a linear mapping. Since *X* is a Gaussian vector, *Y* is also a Gaussian vector.
- One just needs to show that $\mathbb{E}(Y(x)) = 0$ and $Cov(Y(x), Y(y)) = C_h(x, y)$.

► By linearity,
$$\mathbb{E}(Y(x)) = \frac{1}{\sqrt{MN}} (h - m) * \mathbb{E}(X)(x) = 0.$$

• Let $x, y \in \Omega$,

$$Cov(Y(x), Y(y)) = \mathbb{E}(Y(x)Y(y))$$

$$= \frac{1}{MN} \mathbb{E}\left(\sum_{s \in \Omega} (h(s-x) - m)X(s) \sum_{t \in \Omega_{M,N}} (h(t-y) - m)X(t)\right)$$

$$= \frac{1}{MN} \sum_{s,t \in \Omega} (h(s-x) - m)(h(t-y) - m) \sum_{t \in \Omega_{M,N}} \mathbb{E}(X(s)X(t))$$

$$= \frac{1}{MN} \sum_{s \in \Omega} (h(s-x) - m)(h(t-y) - m)$$

$$= C_h(x, y)$$

Differences between *RPN* and *ADSN* Proposition :

- ▶ *RPN* and *ADSN* both have a random phase.
- ► The Fourier modulus of *RPN* is the one of *h*.
- ► The Fourier modulus of *ADSN* is the pointwise multiplication between $|\hat{h}|$ and a Rayleigh noise.



Spot h



RPN Modulus



ADSN Modulus

▶ RPN and ADSN are two different processes.



Spot h



RPN



An ADSN realization



Another ADSN realization

RPN and ADSN associated to texture images

- ▶ We add the original mean to *RPN* and *ADSN* realizations.
- Some textures are relatively well reproduced by *RPN* and *ADSN*.



Original image RPN ADSN

 ... But several developments are necessary to derive texture synthesis algorithms from sample.

◆□▶ ◆□▶ ▲□▶ ▲□▶ □ のQ@

Extension to color images

- We use the RGB color representation for color images.
- ► Color ADSN : The definition of Discrete Spot Noise extends to color images h = (h_r, h_g, h_b).
- ► The color ADSN Y is the limit Gaussian process obtained in letting the number of spots tend to +∞. It is simulated by :

$$Y = \frac{1}{\sqrt{MN}} \begin{pmatrix} (h_r - m_r \mathbf{1}) * X \\ (h_g - m_g \mathbf{1}) * X \\ (h_b - m_b \mathbf{1}) * X \end{pmatrix}, \quad X \text{ a Gaussian white noise.}$$

• One convolves each color channel with the **same** Gaussian white noise *X*.



Phase of color ADSN : The same random phase is added to the Fourier transform of each color channel.

Extension to color images

• **Color** *RPN* : By analogy, the *RPN* associated with a color image $h = (h_r, h_g, h_b)$ is the color image obtained by **adding the same random phase** to the Fourier transform of each color channel.



Extension to color images

Another example with a real-world texture.



Original image h

Color RPN

"Wrong RPN"

- Preserving the original phase displacement between the color channels is essential for color consistency.
- ...however for most monochromatic textures, there is no huge difference.



Original image h

Color RPN

"Wrong *RPN*"

・ コット (雪) (小田) (コット 日)

Avoiding artifacts due to non periodicity

 Both ADSN and RPN algorithms are based on the fast Fourier transform (FFT).

 \implies implicit hypothesis of periodicity

Using non periodic samples yields important artifacts.



Spot h



Avoiding artifacts due to non periodicity

- Our solution : Force the periodicity of the input sample.
- The original image *h* is replaced by its **periodic component** p = per(h), see L. Moisan's course [Moisan, 2011].
- Definition of the periodic component p of h : p unique solution of

$$\begin{cases} \Delta p = \Delta_i h\\ \operatorname{mean}(p) = \operatorname{mean}(h) \end{cases}$$

where, noting N_x the neighborhood of $x \in \Omega$ for 4-connexity :

$$\Delta f(x) = 4f(x) - \sum_{y \in N_x} f(y)$$
 and $\Delta_i f(x) = |N_x \cap \Omega| f(x) - \sum_{y \in N_x \cap \Omega} f(y).$

These two Laplacians only differ at the border :

- Δ : discrete Laplacian with periodic conditions
- Δ_i: discrete Laplacian without periodic conditions (index i for interior)
- *p* is "visually close" to *h* (same Laplacian).
- p is fastly computed using the FFT...

FFT-based Poisson Solver

Periodic Poisson problem : Find the image p such that

$$\begin{cases} \Delta p = \Delta_i h\\ \mathrm{mean}(p) = \mathrm{mean}(h) \end{cases}$$

In the Fourier domain, this system becomes :

$$\begin{cases} \left(4 - 2\cos\left(\frac{2s\pi}{M}\right) - 2\cos\left(\frac{2t\pi}{N}\right)\right)\hat{p}(s,t) = \widehat{\Delta_i h}(s,t), \ (s,t) \in \Omega \setminus \{(0,0)\},\\ \hat{p}(0,0) = \operatorname{mean}(h). \end{cases}$$

Algorithm to compute the periodic component :

- 1. Compute $\Delta_i h$ the discrete Laplacian of *h*.
- 2. Compute m = mean(h).
- 3. Compute $\widehat{\Delta_i h}$ the DFT of $\Delta_i h$ using the forward FFT.
- 4. Compute the DFT \hat{p} of p defined by

$$\begin{cases} \hat{p}(s,t) = \frac{\widehat{\Delta_i h}((s,t))}{-4+2\cos\left(\frac{2s\pi}{M}\right)+2\cos\left(\frac{2t\pi}{N}\right)} & \text{for } (s,t) \neq (0,0)\\ \hat{p}(0,0) = m \end{cases}$$

(日) (日) (日) (日) (日) (日) (日)

5. Compute p using the backward FFT (if necessary).

Periodic component : effects on the Fourier modulus

p is "visually close" to *h* (same Laplacian).

Image h

Periodic component p = per(h)

Smooth component s = h - p (+m)



► The application per : h → p filters out the "cross structure" of the spectrum.

Avoiding artifacts due to non periodicity

Spot h ADSN(h)ADSN(p)

<ロ> <部> <部> <き> <き> <き> <き

Synthesizing textures having arbitrary large size

Ad hoc solution : To synthesize a texture larger than the original spot *h*, one computes an "equivalent spot" \tilde{h} :

- Copy p = per(h) in the center of a constant image equal to the mean of *h*.
- Normalize the variance.
- Attenuate the transition at the inner border.



◆□▶ ◆□▶ ◆□▶ ◆□▶ ● ● ● ●

- Not really rigorous... The envelope changes the covariance.

Properties of the resulting algorithms

- ▶ Both algorithms are fast, with the complexity of the fast Fourier transform $[O(MN \log (MN))].$
- Visual stability : All the realizations obtained from the same input image are visually similar.



Spot h

RPN 1

RPN 2

RPN 3

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

▶ [ON LINE DEMO]

Numerical results : similarity of the textures

In order to compare both algorithms, the same random phase is used for ADSN and RPN.



► Both algorithms produce visually similar textures.

Numerical results : non random phase textures



◆□> ◆□> ◆豆> ◆豆> ・豆 ・ 釣べ⊙

Some other examples of well-reproduced textures...

We only display the RPN result.



Much more examples of success and failures on the IPOL webpage : http://www.ipol.im/pub/algo/ggm_random_phase_texture_ synthesis/

Texton associated with a texture

We work here with gray-level images.

- RPN and ADSN models associated with *h* only depends of the Fourier modulus of *h*.
- Definition : The texton t_h associated with h is the image with the same modulus as h and with zero phase [Desolneux et al, 2012].



- Concentrated in zero : Compact representation of the texture model
- Interesting tool for analysis :

Same texton = same Gaussian texture

Texton for synthesizing textures having arbitrary large size

One computes an extended texton (the texton is smallest at the boundary than the original image) :

$$\tilde{t}_h = m + r(t_h - m) \mathbb{1}_{\Omega}$$



ヘロト 人間 とくほとくほとう

-

Mixing of Gaussian textures

- Using optimal transport distance, one can define barycenters between Gaussian texture models.
- This gives a practical and rigorous solution for Gaussian texture mixing [Xia et al., 2014].



For gray-level images, this is simply obtained in averaging the textons...

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

Conclusion

Summary :

- Random phase noise and asymptotic discrete spot noise have been mathematically defined and theoretically compared.
- Both corresponding texture synthesis algorithms are fast, visually stable, and produce visually similar results.
- Both algorithms reproduce relatively well a certain class of textures : the micro-textures.

Limitations :

- The models are limited to a restrictive class of textures.
- > The algorithms are not robust to non stationarities, perspective effects, ...
- The method is global : the whole texture image has to be computed (in constrast with noise models from computer graphics).

(ロ) (同) (三) (三) (三) (○) (○)

Bibliographic references I

- A. Desolneux, L. Moisan, and S. Ronsin, A compact representation of random phase and Gaussian textures, in ICASSP'12
- B. Galerne, Y. Gousseau, and J.-M. Morel, Random phase textures : Theory and synthesis, IEEE Trans. Image Process., 2011
- B. Galerne, Y. Gousseau, J.-M. Morel, Micro-Texture Synthesis by Phase Randomization, Image Processing On Line, 2011
- D. J. Heeger and J. R. Bergen, Pyramid-based texture analysis/synthesis, SIGGRAPH '95, 1995
- L. Moisan, Periodic plus smooth image decomposition, J. Math. Imag. Vis., 2011
 - - A. V. Oppenheim and J. S. Lim, The importance of phase in signals, Proceedings of the IEEE, 1981
- L.-Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk. State of the art in example-based texture synthesis, Eurographics 2009, 2009.
- J. J. van Wijk, Spot noise texture synthesis for data visualization, SIGGRAPH '91, 1991.
- G.-S. Xia, S. Ferradans, G. Peyre, J-F. Aujol, Synthesizing and mixing stationary Gaussian texture models, SIAM Journal on Imaging Science (SIIMS), 2014.