Modèle gaussien pour la synthèse et l'inpainting de microtextures

Bruno Galerne
bruno.galerne@univ-orleans.fr

Université d'Orléans

Modélisation :
Modèles déterministes et stochastiques pour le traitement d'images
Master de Mathématiques Approfondies

Outline

What is a texture?

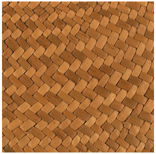
A minimal definition of a **texture** image is an “image containing repeated patterns” [Wei et al., ’09].
The family of patterns reflects a certain amount of randomness, depending on the nature of the texture.
Two main subclasses:

- 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.

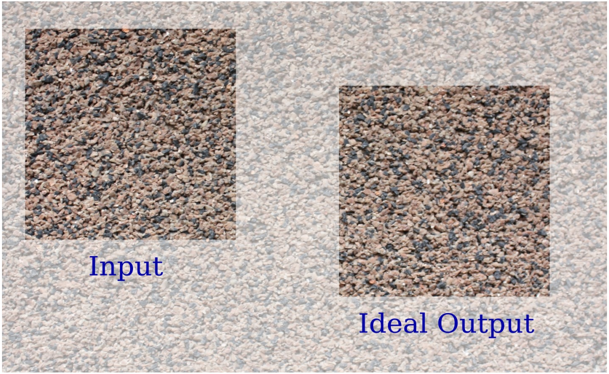

Micro-texture


Macro-texture


Some pebbles

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.

Texture synthesis algorithms

Two main kinds of algorithm:

1. Texture synthesis using statistical constraints:

   Algorithm:

   1. Extract some meaningful “statistics” from the input image (e.g. distribution of colors, of Fourier coefficients, of wavelet coefficients,. . .).
   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

2. 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...
   - See next course (March, 17) for more details.

Texture synthesis algorithms

Two main kinds of algorithm:

1. Texture synthesis using statistical constraints:

   Algorithm:

   1. Extract some meaningful "statistics" from the input image (e.g. distribution of colors, of Fourier coefficients, of wavelet coefficients,. . .).
   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

2. 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...
   - See next course (March, 17) for more details.

Texture synthesis algorithms

Two main kinds of algorithm:

1. Texture synthesis using statistical constraints:
   Algorithm:
   1. Extract some meaningful “statistics” from the input image (e.g. distribution of colors, of Fourier coefficients, of wavelet coefficients,. . .).
   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
2. 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...
   - See next course (March, 17) for more details.

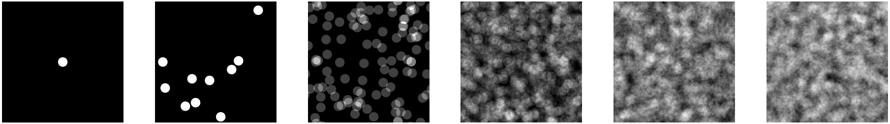
Outline

Circular discrete spot noise [van Wijk, '91]

- Let $h \in \mathbb{R}^{M \times N}$ be a discrete image called *spot* and indexed on the set $\Omega = \{0, \dots, M-1\} \times \{0, \dots, N-1\}$.
- Let (X_k) be a sequence of i.i.d. r.v. uniformly distributed over Ω.
- The **circular discrete spot noise (CDSN) of order** n **associated with** h is the random image

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

where the translations $h(x - X_k)$ are defined periodically.



Spot h | $n = 10$ | $n = 10^2$ | $n = 10^3$ | $n = 10^4$ | $n = 10^5$

Circular asymptotic discrete spot noise (CADSN)

- For texture synthesis we are particularly interested in the limit of the DSN: the ***circular asymptotic discrete spot noise*** **(CADSN)**.
- The CDSN of order n is the sum of the n i.i.d. random images $h(. - X_k)$ $\Longrightarrow$ central limit theorem.

- Definition of CADSN: the CADSN Y associated with h is the limit Gaussian distribution of the normalized discrete spot noise sequence $\left(\frac{f_n - \mathbb{E}(f_n)}{\sqrt{n}}\right)_n$.

- Y is stationary
- The expectation of Y is the mean of h: $\mathbb{E}(Y) = \text{mean}(h) = m$
- The covariance of Y is the autocorrelation of h:

$$\text{Cov}(Y(x), Y(y)) = \frac{1}{MN} \sum_{t \in \Omega} (h(x+t) - m)(h(y+t) - m) = C_h(x - y)$$

Circular asymptotic discrete spot noise (CADSN)

- For texture synthesis we are particularly interested in the limit of the DSN: the ***circular asymptotic discrete spot noise*** **(CADSN)**.
- The CDSN of order n is the sum of the n i.i.d. random images $h(. - X_k)$ $\Longrightarrow$ central limit theorem.

- Definition of CADSN: the CADSN Y associated with h is the limit Gaussian distribution of the normalized discrete spot noise sequence $\left(\frac{f_n - \mathbb{E}(f_n)}{\sqrt{n}}\right)_n$.

- Y is stationary
- The expectation of Y is the mean of h: $\mathbb{E}(Y) = \text{mean}(h) = m$
- The covariance of Y is the autocorrelation of h:

$$\text{Cov}(Y(x), Y(y)) = \frac{1}{MN}\sum_{t\in\Omega}(h(x+t) - m)(h(y+t) - m) = C_h(x - y)$$

Circular asymptotic discrete spot noise (CADSN)

- For texture synthesis we are particularly interested in the limit of the DSN: the ***circular asymptotic discrete spot noise*** **(CADSN)**.
- The CDSN of order n is the sum of the n i.i.d. random images $h(. - X_k)$ $\Longrightarrow$ central limit theorem.

- Definition of CADSN: the CADSN Y associated with h is the limit Gaussian distribution of the normalized discrete spot noise sequence $\left(\frac{f_n - \mathbb{E}(f_n)}{\sqrt{n}}\right)_n$.

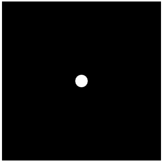
- Y is stationary
- The expectation of Y is the mean of h: $\mathbb{E}(Y) = \mathrm{mean}(h) = m$
- The covariance of Y is the autocorrelation of h:

$$\mathrm{Cov}(Y(x), Y(y)) = \frac{1}{MN} \sum_{t \in \Omega} (h(x+t) - m)(h(y+t) - m) = C_h(x - y)$$

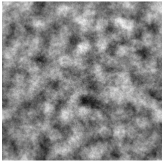
Simulation of the CADSN

Simulation of the CADSN: [Galerne, Gousseau and Morel, '11]

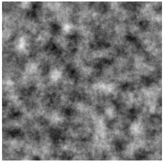
- Let $h \in \mathbb{R}^{M \times N}$ be a an image and X be a Gaussian white noise image.
- The random image $Y = \frac{1}{\sqrt{MN}} (h - \mathrm{mean}(h)) * X$ is the CADSN associated with h.



Spot h



DSN, $n = 10^5$



ADSN

Extension to color images:

- CADSN extends to RGB color images by convolving each color channel by the **same Gaussian white noise** X.

CADSN for Texture Synthesis by Example

- CADSN enables Gaussian texture synthesis by example (once the image is replaced by its periodic component [Moisan, '11]).

Gaussian input		Non Gaussian input	
Image h	CADSN	Image h	CADSN

Interests and limitations

Interests:

- The CADSN reproduces well stationary Gaussian textures, and thus most natural micro-textures.
- CADSN is a fast and reliable algorithm.
- Well-defined mathematical model that as seen several developments:
  - Definition of the canonical texton [Desolneux et al., '12]
  - Gaussian texture mixing using optimal transport barycenter [Xia et al, 2011]
  - **Microtexture inpainting through Gaussian conditional simulation** [Galerne, Leclaire and Moisan, '16] [Galerne, Leclaire, '17a][Galerne, Leclaire, '17b]

Limitations of Gaussian model:

- Gaussian textures are limited: no geometric contours!
- The model is not robust to non stationarities, perspective effects, ...

Limitations due to FFT simulation:

- The method is global: The whole texture image has to be computed.
- It produces periodic images with a fixed size which cannot be extended a posteriori.

Interests and limitations

Interests:

- The CADSN reproduces well stationary Gaussian textures, and thus most natural micro-textures.
- CADSN is a fast and reliable algorithm.
- Well-defined mathematical model that as seen several developments:
  - Definition of the canonical texton [Desolneux et al., '12]
  - Gaussian texture mixing using optimal transport barycenter [Xia et al, 2011]
  - **Microtexture inpainting through Gaussian conditional simulation** [Galerne, Leclaire and Moisan, '16] [Galerne, Leclaire, '17a][Galerne, Leclaire, '17b]

Limitations of Gaussian model:

- Gaussian textures are limited: no geometric contours!
- The model is not robust to non stationarities, perspective effects, ...

Limitations due to FFT simulation:

- The method is global: The whole texture image has to be computed.
- It produces periodic images with a fixed size which cannot be extended a posteriori.

Interests and limitations

Interests:

- The CADSN reproduces well stationary Gaussian textures, and thus most natural micro-textures.
- CADSN is a fast and reliable algorithm.
- Well-defined mathematical model that as seen several developments:
  - Definition of the canonical texton [Desolneux et al., '12]
  - Gaussian texture mixing using optimal transport barycenter [Xia et al, 2011]
  - **Microtexture inpainting through Gaussian conditional simulation** [Galerne, Leclaire and Moisan, '16] [Galerne, Leclaire, '17a][Galerne, Leclaire, '17b]

Limitations of Gaussian model:

- Gaussian textures are limited: no geometric contours!
- The model is not robust to non stationarities, perspective effects, ...

Limitations due to FFT simulation:

- The method is global: The whole texture image has to be computed.
- It produces periodic images with a fixed size which cannot be extended a posteriori.

Interests and limitations

Interests:

- The CADSN reproduces well stationary Gaussian textures, and thus most natural micro-textures.
- CADSN is a fast and reliable algorithm.
- Well-defined mathematical model that as seen several developments:
  - Definition of the canonical texton [Desolneux et al., '12]
  - Gaussian texture mixing using optimal transport barycenter [Xia et al, 2011]
  - **Microtexture inpainting through Gaussian conditional simulation** [Galerne, Leclaire and Moisan, '16] [Galerne, Leclaire, '17a][Galerne, Leclaire, '17b]

Limitations of Gaussian model:

- Gaussian textures are limited: no geometric contours!
- The model is not robust to non stationarities, perspective effects, ...

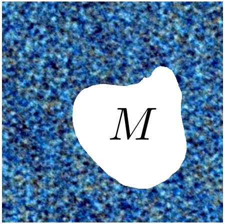
Limitations due to FFT simulation:

- The method is global: The whole texture image has to be computed.
- It produces periodic images with a fixed size which cannot be extended a posteriori.

Outline

Microtexture inpainting

- Inpainting consists in **filling missing regions of an image**.
- In the case of random texture models, inpainting can be formulated as **conditional simulation**
- **Notation:**
  - $\Omega \subset \mathbb{Z}^2$: image domain
  - $M \subset \Omega$: mask
  - u: input texture known only on $\Omega \setminus M$
  - $\mathcal{C}$ a set of conditioning points



Inpainting of a Gaussian texture:

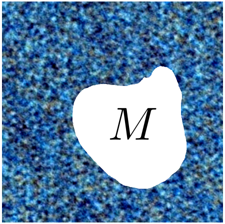
1. Estimation of an ADSN model U from the masked input u.

$$U = \mathrm{moy}(u) + h_u * X \quad \text{where} \quad h_u = \frac{1}{\sqrt{|\Omega \setminus M|}}(u - \mathrm{moy}(u))$$

2. Conditional simulation of U knowing that $U_{|\mathcal{C}} = u_{|\mathcal{C}}$ (using kriging...)

Microtexture inpainting

- Inpainting consists in **filling missing regions of an image**.
- In the case of random texture models, inpainting can be formulated as **conditional simulation**
- **Notation:**
  - $\Omega \subset \mathbb{Z}^2$: image domain
  - $M \subset \Omega$: mask
  - u: input texture known only on $\Omega \setminus M$
  - $\mathcal{C}$ a set of conditioning points



Inpainting of a Gaussian texture:

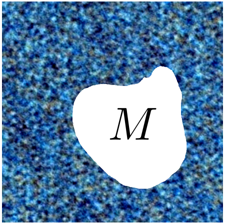
1. Estimation of an ADSN model U from the masked input u.

$$U = \mathrm{moy}(u) + h_u * X \quad \text{where} \quad h_u = \frac{1}{\sqrt{|\Omega \setminus M|}}(u - \mathrm{moy}(u))$$

2. Conditional simulation of U knowing that $U_{|\mathcal{C}} = u_{|\mathcal{C}}$ (using kriging...)

Microtexture inpainting

- Inpainting consists in **filling missing regions of an image**.
- In the case of random texture models, inpainting can be formulated as **conditional simulation**
- **Notation:**
  - $\Omega \subset \mathbb{Z}^2$: image domain
  - $M \subset \Omega$: mask
  - u: input texture known only on $\Omega \setminus M$
  - $\mathcal{C}$ a set of conditioning points



Inpainting of a Gaussian texture:

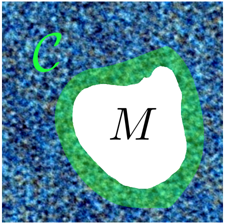
1. Estimation of an ADSN model U from the masked input u.

$$U = \mathrm{moy}(u) + h_u * X \quad \text{where} \quad h_u = \frac{1}{\sqrt{|\Omega \setminus M|}}(u - \mathrm{moy}(u))$$

2. Conditional simulation of U knowing that $U_{|\mathcal{C}} = u_{|\mathcal{C}}$ (using kriging...)

Microtexture inpainting

- Inpainting consists in **filling missing regions of an image**.
- In the case of random texture models, inpainting can be formulated as **conditional simulation**
- **Notation:**
  - $\Omega \subset \mathbb{Z}^2$: image domain
  - $M \subset \Omega$: mask
  - u: input texture known only on $\Omega \setminus M$
  - $\mathcal{C}$ a set of conditioning points



Inpainting of a Gaussian texture:

1. Estimation of an ADSN model U from the masked input u.

$$U = \mathrm{moy}(u) + h_u * X \quad \text{where} \quad h_u = \frac{1}{\sqrt{|\Omega \setminus M|}}(u - \mathrm{moy}(u))$$

2. Conditional simulation of U knowing that $U_{|\mathcal{C}} = u_{|\mathcal{C}}$ (using kriging...)

Gaussian conditional sampling using kriging estimation

- Let $(F(x))_{x\in\Omega}$ be a Gaussian vector **with mean zero** and covariance
$$\Gamma(x,y) = \mathrm{Cov}(F(x),F(y)) = \mathbb{E}(F(x)F(y)), \quad x,y\in\Omega.$$
- The (simple) **kriging estimation** is defined by
$$F^*(x) = \mathbb{E}(\ F(x) \mid F(c),\ c\in\mathcal{C}).$$
- There exists $(\lambda_c(x))_{c\in\mathcal{C}}$ such that $F^*(x) = \sum_{c\in\mathcal{C}} \lambda_c(x)F(c)$.

$\boxed{\text{Theorem: } F^* \text{ and } F - F^* \text{ are independent.}}$ (see e.g. [Lantuéjoul, '02])

Consequence: A conditional sample of F given $F_{|\mathcal{C}} = \varphi$ can be obtained as

$$F \mid F_{|\mathcal{C}} = \varphi \quad \sim \quad \underbrace{\varphi^*}_{\textbf{Kriging component}} \quad + \quad \underbrace{F - F^*}_{\textbf{Innovation component}} \quad .$$

- The **kriging coefficients** $\Lambda = (\lambda_c(x))_{\substack{x\in\Omega\\ c\in\mathcal{C}}}$ satisfy $\Gamma_{|\Omega\times\mathcal{C}} = \Lambda\Gamma_{|\mathcal{C}\times\mathcal{C}}$.
- We use the pseudo-inverse of $\Gamma_{|\mathcal{C}\times\mathcal{C}}$: $\boxed{\Lambda = \Gamma_{|\Omega\times\mathcal{C}}\Gamma^{\dagger}_{|\mathcal{C}\times\mathcal{C}}}$

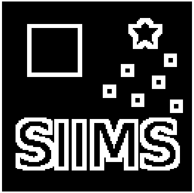
Inpainting of a Gaussian texture

1. Estimation of an ADSN model U from masked input u.
2. Conditional simulation of U knowing that $U_{|\mathcal{C}} = u_{|\mathcal{C}}$:

$$\text{Compute} \quad v = \mathrm{mean}(u) + \underbrace{(u - \mathrm{mean}(u))^{\star}}_{\text{Kriging component}} + \underbrace{U - U^{\star}}_{\text{Innovation component}}$$

Original

Masked input



Conditioning points $\mathcal{C}$

Kriging component

Innovation component

Inpainted texture

Efficient algorithm

- First version presented at ICASSP used explicit matrices to compute

$$\varphi^* = \Gamma_{|\Omega \times \mathcal{C}} \Gamma^{\dagger}_{|\mathcal{C} \times \mathcal{C}} \varphi.$$

- Suitable only for (very) small images !

Scalable Implementation:

- The covariance Γ is the autocorrelation of $h_u = \frac{1}{\sqrt{|\Omega \setminus M|}}(u - \text{moy}(u))$.
- All matrix-vector multiplication with restrictions of Γ can be done using FFT-based convolution.
- Computing $\Gamma^{\dagger}_{|\mathcal{C} \times \mathcal{C}} \varphi$ done using conjugate gradient descent (CGD).
- Each CGD iteration has the cost of a couple of convolutions (and does not depend on the number of points to fill !)
- In practice, 1000 iterations gives a good approximate solution.
- On-line demo with only 100 iterations [Galerne, Leclaire, '17b].
- It turns out that using a 3 pixel wide boundary for $\mathcal{C}$ is visually good enough, and better for the conditioning of the linear system.

Efficient algorithm

- First version presented at ICASSP used explicit matrices to compute

$$\varphi^* = \Gamma_{|\Omega \times \mathcal{C}} \Gamma^{\dagger}_{|\mathcal{C} \times \mathcal{C}} \varphi.$$

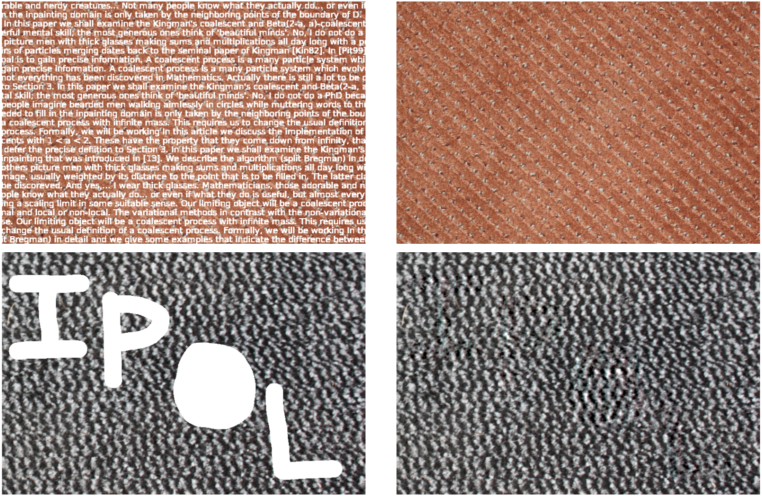
- Suitable only for (very) small images !

Scalable Implementation:

- The covariance Γ is the autocorrelation of $h_u = \frac{1}{\sqrt{|\Omega \setminus M|}}(u - \mathrm{moy}(u))$.
- All matrix-vector multiplication with restrictions of Γ can be done using FFT-based convolution.
- Computing $\Gamma^{\dagger}_{|\mathcal{C} \times \mathcal{C}} \varphi$ done using conjugate gradient descent (CGD).
- Each CGD iteration has the cost of a couple of convolutions (and does not depend on the number of points to fill !)
- In practice, 1000 iterations gives a good approximate solution.
- On-line demo with only 100 iterations [Galerne, Leclaire, '17b].
- It turns out that using a 3 pixel wide boundary for $\mathcal{C}$ is visually good enough, and better for the conditioning of the linear system.

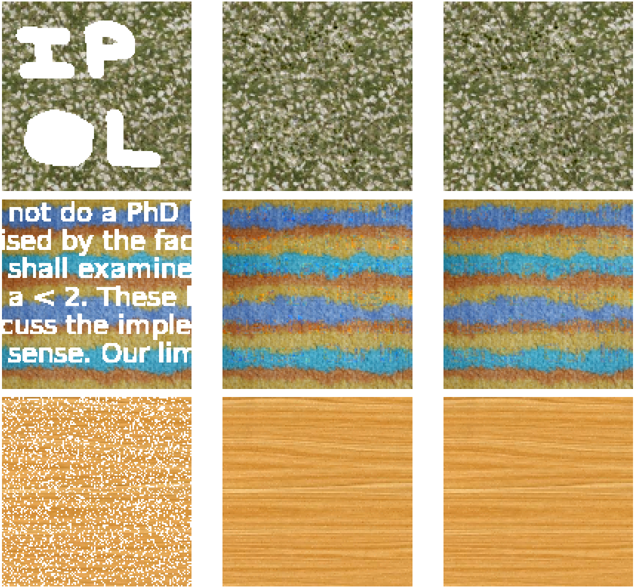
Results: Large problems

Masked texture | Inpainted texture



- Results are satisfying as soon as the Gaussian model is well estimated.

Results: Failures



Input | 100 CGD it. | 1000 CGD it.

Comparison with path-based methods

- Unfair comparison: Other algorithms are **not limited to textures !**

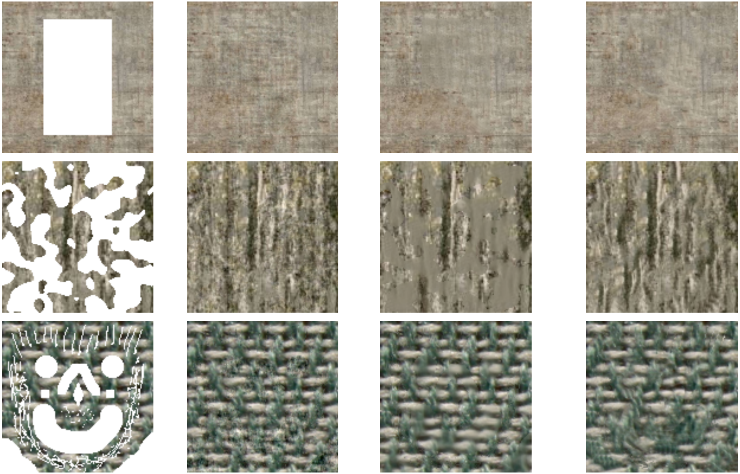
Original | Gaussian inpainting | Kriging component

[Arias et al., ’11] | [Daisy et al., ’15] | [Newson et al., ’14]

- Thanks to the covariance estimation, the Gaussian inpainting is consistent regarding long range correlations.

Comparison with path-based methods

- Our algorithm often gives better results when inpainting a stationary texture, even if the texture is not Gaussian.
- Inpainting textures is not an easy task.



Masked image | Gaussian inp. | [Arias et al., ’11] | [Daisy et al., ’15]

frame I

Bibliographic references

P. Arias, G. Facciolo, V. Caselles, and G. Sapiro,
A variational framework for exemplar-based image inpainting,
International Journal of Computer Vision, 2011.

P. Buyssens, M. Daisy, D. Tschumperlé, and O. Lézoray,
Exemplar-based Inpainting: Technical Review and new Heuristics for better Geometric Reconstructions,
IEEE Transactions on Image Processing, 2015.

R. L. Cook and T. DeRose,
Wavelet noise,
SIGGRAPH '05, ACM, 2005.

A. Desolneux, L. Moisan, and S. Ronsin.
A compact representation of random phase and Gaussian textures.
In ICASSP'12, 2012.

A. A. Efros, and W. T. Freeman.
Image quilting for texture synthesis and transfer.
In SIGGRAPH '01

B. Galerne, Y. Gousseau, and J.-M. Morel,
Random phase textures: Theory and synthesis,
IEEE Trans. Image Proc., 2011.

frame II
Bibliographic references

- B. Galerne, A. Lagae, S. Lefebvre and G. Drettakis,
  Gabor noise by example,
  ACM Trans. Graph. (Proc. of SIGGRAPH 2012), 2012.
- B. Galerne, A. Leclaire and L. Moisan,
  A Texton for Fast and Flexible Gaussian Texture Synthesis,
  EUSIPCO'14, 2014.
- B. Galerne, A. Leclaire, and L. Moisan,
  Microtexture inpainting through Gaussian conditional simulation,
  ICASSP 2016.
- B. Galerne and A. Leclaire.
  Texture Inpainting Using Efficient Gaussian Conditional Simulation.
  SIAM Journal of Imaging Sciences, 2017
- B. Galerne and A. Leclaire.
  An algorithm for gaussian texture inpainting.
  Image Processing On Line, 2017.
- B. Galerne, A. Leclaire, and L. Moisan,
  Texton noise,
  Computer Graphics Forum, 2017.

frame III

Bibliographic references

A. Lagae, S. Lefebvre, G. Drettakis, and P. Dutré,
Procedural noise using sparse Gabor convolution,
SIGGRAPH '09, ACM, 2009.

A. Lagae and S. Lefebvre and R. Cook and T. DeRose and G. Drettakis and D.S. Ebert and J.P. Lewis and K. Perlin and M. Zwicker,
A Survey of Procedural Noise Functions,
Computer Graphics Forum, 2010.

C. Lantuéjoul,
Geostatistical Simulation: Models and Algorithms,
Springer, 2002.

L. Moisan,
Periodic plus smooth image decomposition,
J. Math. Imag. Vis., 2011.

A. Newson, A. Almansa, M. Fradet, Y. Gousseau, and P. Pérez,
Video Inpainting of Complex Scenes,
SIAM Journal on Imaging Sciences, 2014.

K. Perlin,
An Image Synthesizer,
SIGGRAPH '85, ACM, 1985.

frame IV
Bibliographic references

L. Raad and B. Galerne.
Efros and Freeman image quilting algorithm for texture synthesis.
Image Processing On Line, 2017.

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, ACM, 1991.

G.-S. Xia, S. Ferradans, G. Peyré and J.-F. Aujol,
Synthesizing and Mixing Stationary Gaussian Texture Models,
SIAM J. on Imaging Science, 2014.