Published in Image Processing On Line on 2017–10–09.
Submitted on 2016–12–22, accepted on 2017–08–16.
ISSN 2105–1232 © 2017 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2017.198

An Algorithm for Gaussian Texture Inpainting

Bruno Galerne[1], Arthur Leclaire[2]

[1] Laboratoire MAP5, Université Paris Descartes and CNRS, Sorbonne Paris Cité, France
(bruno.galerne@parisdescartes.fr)
[2] CMLA, ENS Cachan, CNRS, Université Paris-Saclay, 94235 Cachan, France
(arthur.leclaire@cmla.ens-cachan.fr)

Communicated by José Lezama *Demo edited by* Arthur Leclaire

This IPOL article is related to a companion publication in the SIAM Journal on Imaging Sciences:
Bruno Galerne, Arthur Leclaire, "Texture Inpainting Using Efficient Gaussian Conditional Simulation" *SIAM Journal on Imaging Sciences*, vol. 10, no. 3, pp. 1446–1474, 2017. https://doi.org/10.1137/16M1109047

Abstract

Inpainting consists in computing a plausible completion of missing parts of an image given the available content. In the case of images composed of a homogeneous microtexture, inpainting can be addressed by relying on Gaussian conditional simulation. In this paper we describe an algorithm which allows to perform inpainting by Gaussian conditional simulation, in a scalable way. We provide a detailed numerical study of this algorithm.

Source Code

The reviewed source code and documentation for this algorithm are available from the web page of this article[1]. Compilation and usage instruction are included in the `README.txt` file of the archive.

Supplementary Material

A Matlab interface (not reviewed) is also available for convenience.

Keywords: inpainting; Gaussian textures; conditional simulation; simple kriging

[1] https://doi.org/10.5201/ipol.2017.198

1 Introduction

Inpainting consists in filling missing regions in images by inferring from the surrounding context. It was addressed by early methods in [17, 4, 16] which, to some extent, fill the hole by connecting the incoming level lines in order to satisfy the Gestaltist' principle of good continuation. In the spirit of these seminal works, many contributions have addressed geometric inpainting, and we refer to [19] for a detailed review of these methods.

Unfortunately, these geometric inpainting methods, which are essentially based on PDEs or generic variational problems, are unable to properly model the stochastic behavior of textures, and thus fail to inpaint textural parts of images. In order to inpaint textural content, one can estimate from the surrounding context a texture model that is conditionally sampled knowing the values on the boundary of the hole. In turn, the exemplar-based texture synthesis methods of [7, 20] can be modified in order to address textural inpainting: these articles indeed contain a few inpainting results under the name "constrained texture synthesis". Many inpainting methods were inspired by this exemplar-based algorithm, leading to state-of-the-art inpainting results [21, 3, 2, 1, 15, 12, 18, 5]. But still, as for texture synthesis, these patch-based methods do not provide mathematical guarantees on the inpainted content (except the asymptotic result of [14]).

In this paper we propose an algorithm (together with an online implementation) which is based on Gaussian conditional simulation, and thoroughly described in the companion paper. The first step of the method is to estimate a Gaussian texture model on the masked exemplar. Next, this Gaussian model is conditionally sampled using the available values of the texture on the outer border of the inpainting domain. This conditional sampling step essentially amounts to solve a possibly very large and ill-conditioned linear system. We propose to solve this linear system by relying on a conjugate gradient descent (CGD) which takes profit on a Fourier-based implementation of the system matrix, computed with the fast Fourier transform (FFT). We provide a detailed numerical study of this algorithm, and several inpainting results on stochastic and structured textures, showing that this algorithm competes with state-of-the-art methods on irregular textures.

2 Detailed Description of the Algorithm

Let $\Omega \subset \mathbb{Z}^2$ be a $m \times n$ rectangle of $\mathbb{Z}^2$. Let $u : \Omega \to \mathbb{R}^d$ be the image to inpaint; d is the number of channels ($d = 3$ for color images). Let $M \subset \Omega$ be the indices corresponding to masked pixels, so that the values of u are known only on $\Omega \setminus M$. We also define $\mathcal{C} = \partial_w M$ the outer border of M with width w pixels, which is composed of all pixels of $\Omega \setminus M$ which are at Euclidean distance $\leq w$ of M. If $S \subset \Omega$, we will denote by $u_{|S}$ the restriction of u to S.

Our algorithm consists in first estimating a Gaussian texture model U using the values of u outside the mask, and next performing conditional simulation of U knowing that $U_{|\mathcal{C}} = u_{|\mathcal{C}}$.

2.1 Estimation of the Gaussian Model

We set $\omega = \Omega \setminus M$ and we set $v = u_{|\omega}$ the restriction of u to ω. When inpainting composite images, other (manual) choices of ω can be interesting; but in this online demo, we only consider the canonical choice $\omega = \Omega \setminus M$ which suffices to inpaint images composed of one microtexture.

Following [8, 10], we estimate a Gaussian texture model with

$$\bar{v} = \frac{1}{|\omega|} \sum_{x \in \omega} v(x), \qquad \text{and} \quad \forall x \in \mathbb{Z}^2,\ t_v(x) = \begin{cases} \frac{1}{\sqrt{|\omega|}}(v(x) - \bar{v}) & \text{if } x \in \omega, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The function t_v is called the texton associated to v. The Gaussian texture model is then given by

$$\forall x \in \Omega, \quad U(x) = \bar{v} + t_v * W(x) = \bar{v} + \sum_{y \in \mathbb{Z}^2} t_v(x-y)W(y), \tag{2}$$

where W is a normalized Gaussian white noise on $\mathbb{Z}^2$ (i.e. $(W(x))_{x\in\mathbb{Z}^2}$ is a collection of i.i.d. Gaussian random variables with mean 0 and standard deviation 1). The second term is the convolution of t_v with the scalar Gaussian white noise W. Notice that t_v has a finite support so that this convolution can be computed in Fourier domain, provided that t_v is extended by zero to a twice larger domain. Notice also that this formula holds in the color case: each channel of t_v is convolved with the same white noise W.

U is a $\mathbb{R}^d$-valued stationary Gaussian random field on Ω. The mean of U is constant equal to $\bar{v}$ and its covariance Γ is given by

$$\forall x, y \in \Omega, \quad \Gamma(x,y) = \mathbb{E}\Big((U(x)-\bar{v})(U(y)-\bar{v})^T\Big) = \sum_{z\in\mathbb{Z}^2} t_v(x-z)t_v^T(y-z). \tag{3}$$

Since U is a $\mathbb{R}^d$-valued random field, $\Gamma(x,y)$ is a $d\times d$-matrix. Alternately, by writing also the color index, one can also consider that Γ is a $|\Omega\times\{1,\dots,d\}|\times|\Omega\times\{1,\dots,d\}|$ matrix. But in practice we do not form this matrix explicitly since for conditional simulation (Subsection 2.2) we will only need to apply the operator associated to the covariance, which can be done as follows. This operator is the convolution by the matrix kernel

$$t_v * \widetilde{t}_v^T(h) = \sum_{z\in\mathbb{Z}^2} t_v(z)\widetilde{t}_v^T(h-z), = \sum_{z\in\mathbb{Z}^2} t_v(z)t_v^T(z-h) \tag{4}$$

where we used the notation $\widetilde{t}(x) = t(-x)$ for the mirror function. Indeed, since

$$\Gamma(x,y) = t_v * \widetilde{t}_v^T(x-y), \tag{5}$$

we have for all images $\varphi \in (\mathbb{R}^d)^\Omega$,

$$\sum_{y\in\Omega} \Gamma(x,y)\varphi(y) = \sum_{y\in\Omega} t_v * \widetilde{t}_v(x-y)^T\varphi(y) = t_v * \widetilde{t}_v^T * \varphi(x). \tag{6}$$

Again, this convolution can be computed in Fourier domain (extending the image φ by zero-padding). One can also see that this formula naturally encompasses the color case because the matrix kernel $t_v * \widetilde{t}_v^T$ contains the cross-correlations between all color channels of v.

In order to give a description which is closer to the provided code, we summarize the model estimation, the model sampling and the covariance operator in Algorithm 1, Algorithm 2 and Algorithm 3, respectively. In these algorithms, we denote by

$$Z : (\mathbb{R}^d)^{m\times n} \longrightarrow (\mathbb{R}^d)^{2m\times 2n}, \tag{7}$$

the zero-padding operator which extends a $m\times n$ image by zero to a domain of size $2m\times 2n$. Its transpose Z^T is a restriction operator. Also, we denote by

$$K : \mathbb{R}^{2m\times 2n} \longrightarrow (\mathbb{R}^d)^{2m\times 2n}, \tag{8}$$

the periodic convolution of a real-valued image of size $2m\times 2n$ by the color image $Z(t_v)$ (this convolution can be computed with a FFT of size $2m\times 2n$). Its transpose K^T is the periodic convolution by $\widetilde{Z(t_v^T)}$. With this notation, a sample of the Gaussian model is given by $\bar{v} + Z^TKW$ where W is a Gaussian white noise of size $2m\times 2n$, and the covariance operator can thus be written Z^TKK^TZ.

Algorithm 1: Estimating the Gaussian model

input : masked exemplar u, mask M

$\omega \leftarrow \Omega \setminus M$

Compute the restriction $v \leftarrow u_{|\omega}$.

$$\bar{v} \leftarrow \frac{1}{|\omega|}\sum_{x\in\omega} v(x), \qquad t_v \leftarrow \frac{1}{\sqrt{|\omega|}}(v-\bar{v})\mathbf{1}_\omega \qquad (\mathbf{1}_\omega \text{ is the indicator function of } \omega)$$

output: Gaussian model $\mathcal{N}(\bar{v}, \Gamma)$.

The matrix Γ *is defined by*

$$\forall x, y \in \Omega, \quad \Gamma(x,y) = t_v * \widetilde{t_v^T}(x-y)$$

but not stored explicitly.

Algorithm 2: Sample the Gaussian model (2) on Ω

input : texton t_v, mean value $\bar{v}$

Draw a normalized Gaussian white noise W of size $2m \times 2n$.

Perform periodic convolution of W with $Z(t_v)$. *(with the FFT), (Z defined by (7))*

Crop the restriction on Ω.

Add the mean value $\bar{v}$.

Compute $U = \bar{v} + Z^T K W$. *(Z defined by (7), K defined by (8))*

output: sample U of the Gaussian model $\mathcal{N}(\bar{v}, \Gamma)$, ($\Gamma$ defined by (3)).

Algorithm 3: Apply the covariance operator $\Gamma = Z^T K K^T Z$

input : texton t_v, test image $\chi \in (\mathbb{R}^d)^\Omega$

Extend χ by zero-padding to a twice-larger domain.

Perform periodic convolution with $\widetilde{Z(t_v)}$. *(with the FFT), (Z defined by (7))*

Perform periodic convolution with $\widetilde{Z(t_v^T)}$. *(with the FFT), (Z defined by (7))*

Crop the result on Ω.

output: $\Gamma\chi$ (image of χ by the convolution operator Γ).

2.2 Conditional Simulation

In this paragraph, we explain how to conditionally sample the estimated Gaussian model given the value on the conditioning points $\mathcal{C}$. Withdrawing the mean component, this is equivalent to conditionally sample the random field $F = t_v * W$ knowing $F_{|\mathcal{C}} = u_{|\mathcal{C}} - \bar{v}$.

In order to clarify the notation we introduce the restriction operator $R : (\mathbb{R}^d)^\Omega \to (\mathbb{R}^d)^\mathcal{C}$ that keeps only the values on $\mathcal{C}$. Its transpose $R^T : (\mathbb{R}^d)^\mathcal{C} \to (\mathbb{R}^d)^\Omega$ is a zero-padding operator. Using the traditional algorithm for Gaussian conditional simulation [9], we see that a conditional sample of F given $RF = \varphi$ can be obtained from an unconditional sample $F \sim \mathcal{N}(0, \Gamma)$ by computing

$$F + \Lambda(\varphi - RF), \tag{9}$$

where $\Lambda = \Gamma R^T (R\Gamma R^T)^\dagger$ (whose elements are sometimes called the kriging coefficients). The notation $A^\dagger$ refers to the pseudo-inverse of A. For more details about the derivation of this formula, we refer to the SIIMS companion paper [9]. Once a sample of F has been obtained (using the technique explained in the previous section), the main difficulty is thus to apply the matrix Λ. The multiplication by ΓR^T is easy: it consists in zero-padding from $\mathcal{C}$ to Ω and applying the covariance operator Γ (applied with Algorithm 3). Computing $A^\dagger\varphi$ where $A = R\Gamma R^T$ is more costly. Let us notice that $R\Gamma R^T$ is

the covariance of $R(F)$ and thus applying A can be done efficiently, see Algorithm 4. But computing the pseudo-inverse is less easy.

Algorithm 4: Apply the restricted covariance operator $A = R\Gamma R^T$

input : texton t_v, subdomain $\mathcal{C}$, test image $\psi \in (\mathbb{R}^d)^\Omega$
Extend ψ to Ω by zero-padding.
Apply the covariance operator Γ. *(with Algorithm 3)*
Extract the values on $\mathcal{C}$.
output: $A\psi$ (image of ψ by the restricted covariance operator A)

Assume for a moment that A is invertible. Then computing $A^{-1}\varphi$ amounts to solving a linear system of size $p \times p$ (where $p = d|\mathcal{C}|$). Since A is symmetric positive-definite, this can be reduced to solving two triangular systems thanks to the Cholesky factorization of A. Nevertheless, finding the Cholesky factorization of A requires $\mathcal{O}(p^3)$ flops in general. Therefore, this direct method will only work for small values of p. This was a major limitation of our preliminary work presented in [11].

To cope with this problem, we propose here to solve the linear system with a conjugate gradient descent (CGD) algorithm, taking profit of the fact that applying the matrix A can be done efficiently. Therefore, following [13], we compute $A^\dagger\varphi$ by performing a CGD on

$$f : \psi \longmapsto \frac{1}{2}\|A\psi - \varphi\|^2, \tag{10}$$

with initialization $\psi_0 = 0$. This optimization procedure solves the normal equations $A^TA\psi = A^T\varphi$, which are equivalent to $A\psi = \varphi$ when $\varphi \in \mathrm{Range}(A)$ (recall that the range of A and the kernel of A^T are orthogonal subspaces). Besides, using the CGD on the normal equations allows to cope with possibly singular matrices A. This algorithm is summarized in Algorithm 5.

Algorithm 5: Conjugate gradient descent to compute $A^\dagger\varphi$

input : A, φ, precision $\varepsilon > 0$, max number of iterations $k_{\max}$
initialization: $k \leftarrow 0$, $\psi_0 \leftarrow 0$, $r_0 \leftarrow A^T\varphi - A^TA\psi_0$, $d_0 \leftarrow r_0$.
while $\|r_k\| > \varepsilon$ *and* $k < k_{\max}$ **do**
  $k \leftarrow k + 1$
  $\alpha_k \leftarrow \frac{\|r_k\|^2}{d_k^T A^T A d_k}$ *(Apply A with Algorithm 4)*
  $\psi_{k+1} \leftarrow \psi_k + \alpha_k d_k$
  $r_{k+1} \leftarrow r_k - \alpha_k A^T A d_k$
  $d_{k+1} \leftarrow r_{k+1} + \frac{\|r_{k+1}\|^2}{\|r_k\|^2} d_k$
output: approximation of $A^\dagger\varphi$ given by ψ_k

Stopping criterion

The stopping criterion that we use in Algorithm 5 is $\|r_k\| \leq \varepsilon$ where the residual at iteration k is given by

$$r_k = A^T\varphi - A^TA\psi_k, \tag{11}$$

and where $\|r_k\|$ is the unnormalized ℓ^2-norm of $r_k \in \mathbb{R}^{|\mathcal{C}|}$. The behavior of the residual norm along the iterations will be studied in Section 3. In practice, to keep a simple choice, we take $\varepsilon := 10^{-3}$ and we also constrain the number of iterations to be less than $k_{\max} = 1000$.

2.3 Complete Algorithm

We summarize the whole microtexture inpainting algorithm in Algorithm 6. The overall complexity of this algorithm is $\mathcal{O}(k_{\max}|\Omega|\log|\Omega|)$ where $k_{\max}$ is the number of iterations used in the gradient descent algorithm. More precisely, let us evaluate the precise number of calls to the FFT. The Fourier transform of the texton is computed once and for all with d FFTs (where d is the number of channels). Sampling the Gaussian model requires d FFTs (the white noise can be sampled directly in Fourier domain). Initializing the CGD requires to apply the matrix A which costs $2d$ FFTs (d direct and d inverse). Each iteration of the CGD requires to apply twice the matrix A. The last convolution with the covariance operator requires $2d$ FFTs. Eventually, the number of FFTs required by the whole inpainting process is $(4k_{\max}+6)d$ FFTs. Using our C implementation (involving parallel computing, in particular for the FFT) run with a modern computer (Intel i7 processor @2.60GHz), the whole inpainting process takes about 20 seconds for a 256×256 image and 1000 iterations of CGD. When using this algorithm, we recommend to keep $k_{max}=1000$ iterations. However, since we want the IPOL implementation to be able to inpaint possibly large images, we set by default $k_{\max}=100$ on the online demo (leaving $k_{\max}$ as a parameter that can be changed on the webpage).

Algorithm 6: Microtexture inpainting

input : mask $M\subset\Omega$, texture u on $\Omega\setminus M$, conditioning points $\mathcal{C}=\partial_3 M$

notation: R denotes the operator associated to the restriction on $\mathcal{C}$

From the restriction v of u to $\omega=\Omega\setminus M$, estimate a Gaussian model $\mathcal{N}(\bar{v},\Gamma)$ with

$$\bar{v}\leftarrow\frac{1}{|\omega|}\sum_{x\in\omega}v(x),\quad t_v\leftarrow\frac{1}{\sqrt{|\omega|}}(v-\bar{v})\mathbf{1}_\omega$$ *(Algorithm 1), ($\mathbf{1}_\omega$ is the indicator function of ω)*

Sample the Gaussian model $F=Z^TKW$, where $W\sim\mathcal{N}(0,I)$ *(Algorithm 2)*

$\psi\leftarrow(R\Gamma R^T)^\dagger(Ru-\bar{v}-RF)$ *(Algorithm 5 with $A=R\Gamma R^T$, $\varepsilon=10^{-3}$ and $k_{\max}=1000$ iterations)*

Compute $\Lambda(Ru-\bar{v}-RF)=\Gamma R^T\psi$ *(Zero-padding and Algorithm 3)*

$w\leftarrow\bar{v}+F+\Lambda(Ru-\bar{v}-RF)$

Force the initial values outside the mask $w_{|\Omega\setminus M}\leftarrow u_{|\Omega\setminus M}$

output: inpainted image w

3 Numerical Study of the Conjugate Gradient Descent

3.1 Possible Regularization

For our inpainting application, it is often harmless to modify the texture model by adding a Gaussian white noise of variance δ^2, provided that δ^2 is small compared to the color variance in the observed texture. This is equivalent to replace the matrix Γ by $\Gamma+\delta^2 I$. The matrix $A=\Gamma_{|\mathcal{C}\times\mathcal{C}}$ is modified accordingly as $A+\delta^2 I$. With this modification, the matrix A is ensured to be invertible; in other words, it adds some regularization in the linear system. Besides, if λ is an eigenvalue of A, then $\lambda+\delta^2 I$ is an eigenvalue of $A+\delta^2 I$. Therefore, adding a small white noise tends to decrease the condition number of A. As illustrated in Figure 1, this may improve the convergence properties of the algorithm, but it does not improve the visual results obtained after $k=1000$ iterations. So by default, we do not use such regularization.

3.2 Precision on the Solution

The error on the computation of $A^\dagger\varphi$ made by stopping at iteration k is given by $A^\dagger\varphi-\psi_k$, which is connected to the residual via $r_k=A^TA(A^\dagger\varphi-\psi_k)$. Thus, the error norm is $\|A^\dagger\varphi-\psi_k\|=\|(A^TA)^\dagger r_k\|$.

Once $A^\dagger\varphi$ is computed, the conditional sample involves the vector $\Gamma_{|\Omega\times\mathcal{C}}A^\dagger\varphi$. If we replace $A^\dagger\varphi$ by the approximation ψ_k, we thus make an error given by

$$\|\Gamma_{|\Omega\times\mathcal{C}}(A^\dagger\varphi-\psi_k)\| = \|\Gamma_{|\Omega\times\mathcal{C}}(A^TA)^\dagger r_k\| \leq \|\Gamma_{|\Omega\times\mathcal{C}}\|\|(A^TA)^\dagger\|\|r_k\|, \tag{12}$$

where the matrix norms are derived from the current norm $\|\cdot\|$. If we want an error norm $< \eta$ on the inpainting output, we should use a stopping criterion

$$\|r_k\| < \varepsilon := \frac{\eta}{\|\Gamma_{|\Omega\times\mathcal{C}}\|\|(A^TA)^\dagger\|}. \tag{13}$$

In this case of the ℓ^2-norm, $\|\Gamma_{|\Omega\times\mathcal{C}}\| \leq \|\Gamma\|$ which is the largest eigenvalue of Γ. If Γ is the covariance of a Gaussian model estimated on u, the discrete Fourier basis is an eigenvector basis of Γ, and thus $\|\Gamma_{|\Omega\times\mathcal{C}}\| \leq \max_\xi |\widehat{t_u}(\xi)|^2$, where $\widehat{t_u}$ is the DFT of t_u. In a standard inpainting experiment, numerical computations give a value $\|\Gamma_{|\Omega\times\mathcal{C}}\| \leq 100$. In contrast, the second matrix norm is much larger. Indeed, $\|(A^TA)^\dagger\| = \lambda^{-2}$ where λ is the smallest non-zero eigenvalue of A. In some standard inpainting experiment, we obtained a λ value close to 10^{-4} which makes the stopping criterion of (13) impractical.

Therefore, in practice, we cannot use a stopping criterion $\|A^\dagger\varphi-\psi_k\| < \eta$ on the error, but instead we use a stopping criterion directly on the residual $\|r_k\| < \varepsilon = 10^{-3}$. We also constrain the number of iterations to be less than $k_{\max} < 1000$ to keep the algorithm scalable. Indeed, the decreasing speed of the residual norm strongly depends on the size of the system, which is directly related to the size of the conditioning set (for big holes in images much larger than those shown in Figure 1, 1000 iterations may not suffice to get $\|r_k\| < 10^{-3}$).

3.3 Convergence of the Conjugate Gradient Algorithm

In this paragraph, we examine the behavior of Algorithm 5 on a simple inpainting case with a small image. Precisely, we consider a 64×64 image with $d = 3$ color channels, on which we put a mask of size 11×11 and we take $\mathcal{C} = \partial_3 M$. In such a case, the matrices $A = \Gamma_{|\mathcal{C}\times\mathcal{C}}$ and $\Lambda = \Gamma_{|\Omega\times\mathcal{C}}\Gamma^\dagger_{|\mathcal{C}\times\mathcal{C}}$ can be formed explicitly, and the matrix A is invertible (though ill conditioned; its condition number is $> 10^6$). Recall that in the RGB case, all the channels correlations must be considered, so that A is actually a $d|\mathcal{C}| \times d|\mathcal{C}|$ matrix with $d|\mathcal{C}| = 504$.

As a baseline comparison, we solve the linear system $A\psi = \varphi$ with a standard matrix method based on the Cholesky decomposition of A and we use this reference solution ψ_{ref} to examine the convergence speed of the conjugate gradient method applied on the normal equations. We run this algorithm for $k_{\max} = 10000$ iterations and we monitor the evolution of

$$\text{err}(k) = \frac{1}{\sqrt{d|M|}}\|\Gamma_{|M\times\mathcal{C}}\psi_{\text{ref}} - \Gamma_{|M\times\mathcal{C}}\psi_k\|_2, \tag{14}$$

$$\text{resn}(k) = \frac{1}{\sqrt{d|\mathcal{C}|}}\|A^T\varphi - A^TA\psi_k\|_2, \tag{15}$$

the former being the normalized ℓ^2 error on the inpainting result made by stopping at iteration k, and the latter being the normalized ℓ^2-norm of the residual at iteration k.

These numerical results are reported in Figure 1. As one can see on these graphs, even if the residual norm quickly becomes $< 10^{-4}$, a very large number of iterations is necessary to attain a value close to machine precision. Also, one can notice that the convergence to the ideal solution is quite slow, but still, the result obtained after 10000 iterations remains a satisfying inpainting result. In particular, in terms of visual quality of the inpainting (which is, of course, a subjective measure)

the solution obtained after 10000 iterations is very close to the reference. In Section 4, we will even see that the result after $k_{\max} = 1000$ or even $k_{\max} = 100$ iterations is still satisfying in general.

In Figure 1, we also show the numerical results obtained by adding some regularization in the linear system as suggested above. More precisely, we replaced Γ by $\Gamma + \delta^2 I$. In the experiment shown here, we take $\delta = 0.01$. As it is well known, adding such a regularization tends to decrease the condition number of the system (which is ≈ 3430 here after regularization), and thus strongly improves the convergence speed of the gradient descent. In particular, after 5000 iterations, we get an inpainting error $\approx 10^{-14}$, which is the best we can hope since we are limited by the numerical precision of the FFT algorithm. Let us also remark that the corresponding inpainting result is visually close to the one obtained without regularization.

To end this convergence study, we illustrate in Figure 2 a more difficult case: we inpaint the same image, but now taking $\mathcal{C} = \Omega \setminus M$. The corresponding linear system is now $d|\mathcal{C}| \times d|\mathcal{C}|$ where $d|\mathcal{C}| = 11925$; in particular we have $d|\mathcal{C}| \geq |\Omega|$ and thus A is necessarily singular (see the companion paper [9, §3.4]). In this case, we do not dispose of a reference solution to compute the pseudo-inverse $A^\dagger$, so we can only illustrate the convergence with the ℓ^2 norm of the residual. As one can see in Figure 2, the convergence is still slow in this case, and adding a small regularization ($\delta = 0.01$) does not help much. But in terms of visual result, the inpainting is already satisfying even if the algorithm has not fully converged. Of course, increasing further the regularization δ improves the convergence speed, but also impacts the model more significantly (i.e. Gaussian noise becomes visible in the inpainted part). This is why we choose not to include regularization by default.

To conclude, the conjugate gradient descent on normal equations is a principled and scalable way to compute the solution of the kriging system. It may converge quite slowly in general, but gives visually satisfying inpainting results before full convergence.

4 Experiments

4.1 Inpainting Examples

In Figure 3, Figure 4 and Figure 5 we display inpainting results obtained with several textures and different kinds of masks. As one can see, our algorithm is able to inpaint perfectly microtexture images, provided that we dispose of a sufficiently plain piece of texture to properly estimate the Gaussian model. If the mask is too scattered and irregular (like in the last row of Figure 4) then the estimation step gives a too imprecise result; but still, the inpainting algorithm can still give a reasonable result (last rows of Figures 4 and 5).

Since this inpainting algorithm relies on a Gaussian texture model, it is more adapted to microtextures than structured textures. Nevertheless, this algorithm is able to produce interesting inpainting results on structured textures, especially with sufficiently thin masks (see the first examples of Figure 4 and the right column of Figure 5). In particular, the results on such textures may look much more plausible than results obtained with methods based on PDEs or generic variational methods (like TV inpainting [6]).

We have discussed the number of iterations from a numerical point of view in Section 3. In Figure 3, Figure 4 and Figure 5, we observe that the results obtained after 100 or 1000 iterations (in the CGD) are hardly distinguishable when compared side by side. However, let us mention that iterating enough is important to enforce the constraints at the boundary of the inpainting domain. For example, in the last row of Figure 4, the boundary of the mask is less visible after 1000 iterations. In general, we recommend to run the algorithm for 1000 iterations to ensure a good inpainting result. But for the online demo, we set the default number of iterations to 100 so that the demo scales to very large images (the user is free to increase this parameter if needed).

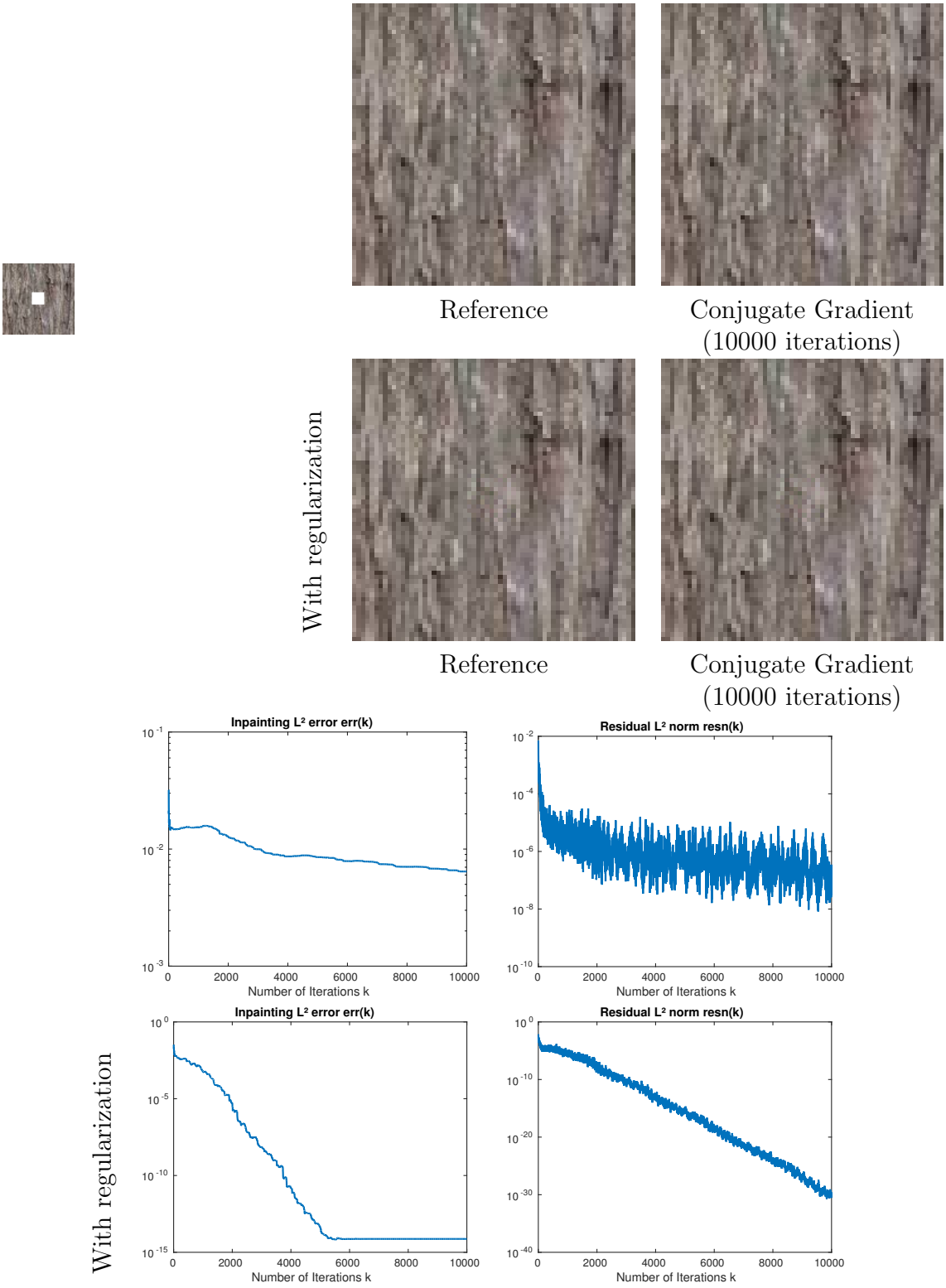


Figure 1: **Convergence of the conjugate gradient method.** $\mathcal{C} = \partial_3 M$**.** On the first row, from left to right, we display a masked exemplar texture, the reference inpainting result (obtained by computing the reference solution ψ_{ref} with a matrix method based on Cholesky decomposition), and the inpainted result obtained with our FFT based algorithm (using $k_{\max} = 10000$ iterations of the conjugate gradient method on the normal equations). The conditioning points $\mathcal{C}$ are taken on a 3 pixel wide border of the mask. On the second row, we show the evolution along the iterations of the inpainting normalized ℓ^2 error (defined by (14)) and the normalized ℓ^2-norm of the residual (defined by (15)). The inpainting error is computed with respect to the reference. These graphs indicate that the convergence of the algorithm is in general very slow, but gets better when adding some regularization as suggested in Section 3.1. See the text for additional comments.

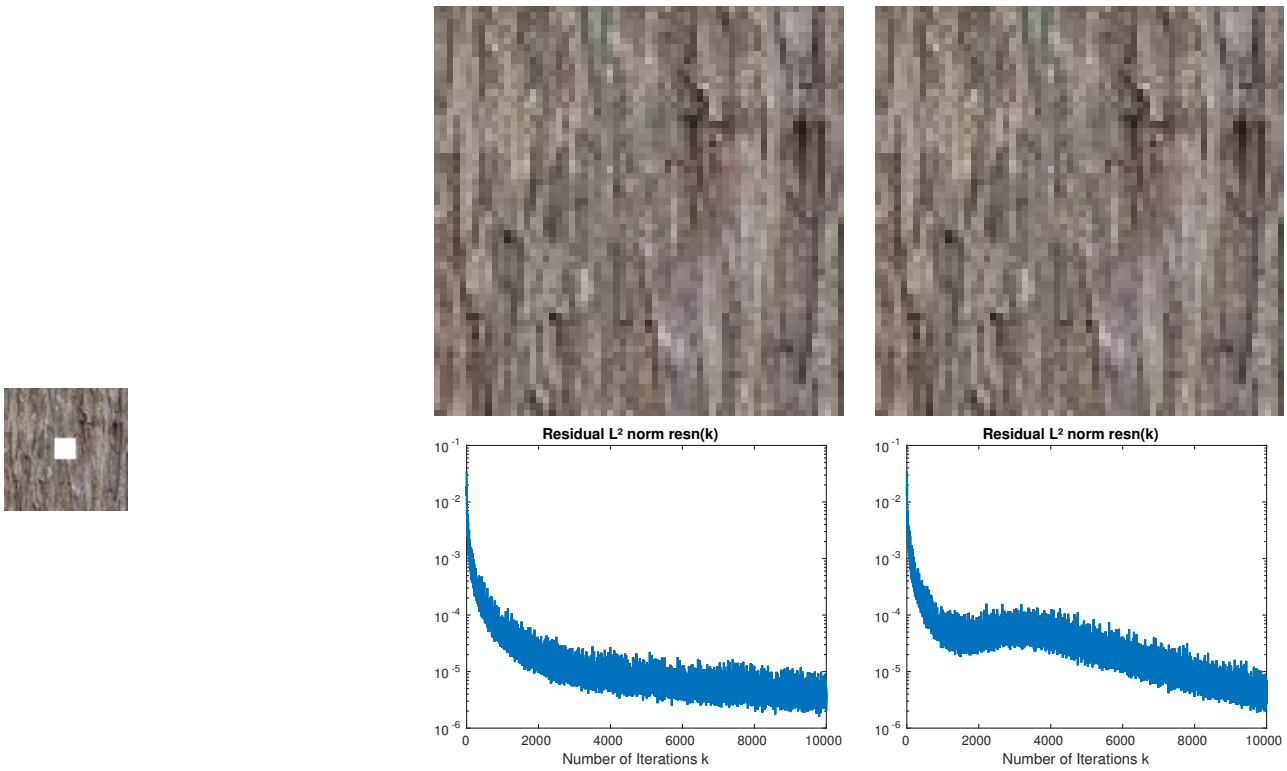
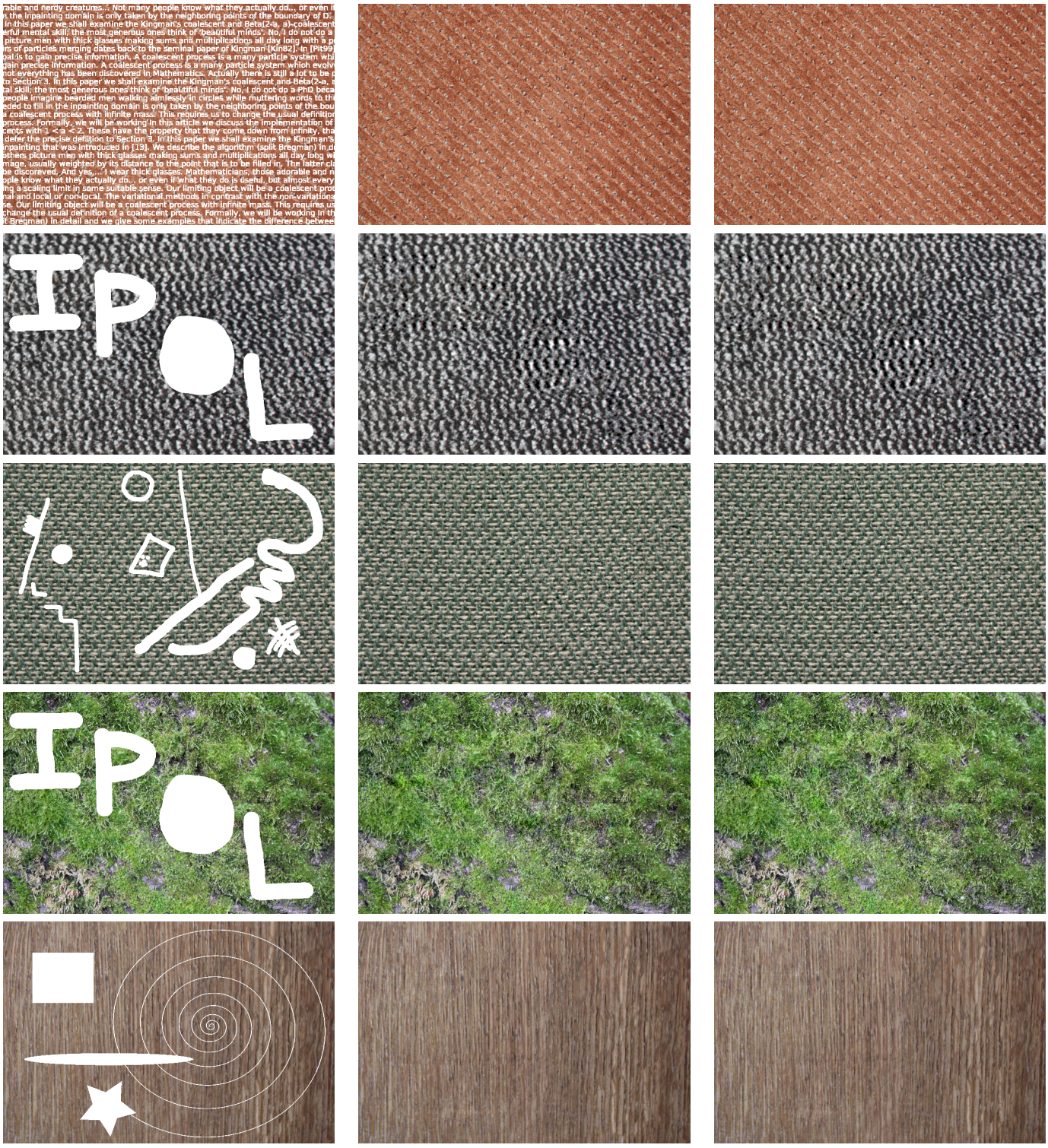


Figure 2: **Convergence of the conjugate gradient method.** $\mathcal{C} = \Omega \setminus M$**.** The first column contains the masked exemplar. The second and third columns contain the inpainting results without and with regularization, respectively. The inpainting results have been obtained with $k_{\max} = 10000$ iterations of the conjugate gradient on normal equations. Again, the residual norm decreases to 0 quite slowly. The regularization parameter $\delta = 0.01$ is not sufficient to improve the convergence. Still, the inpainting results are already satisfying, even without regularization. See the text for additional comments.

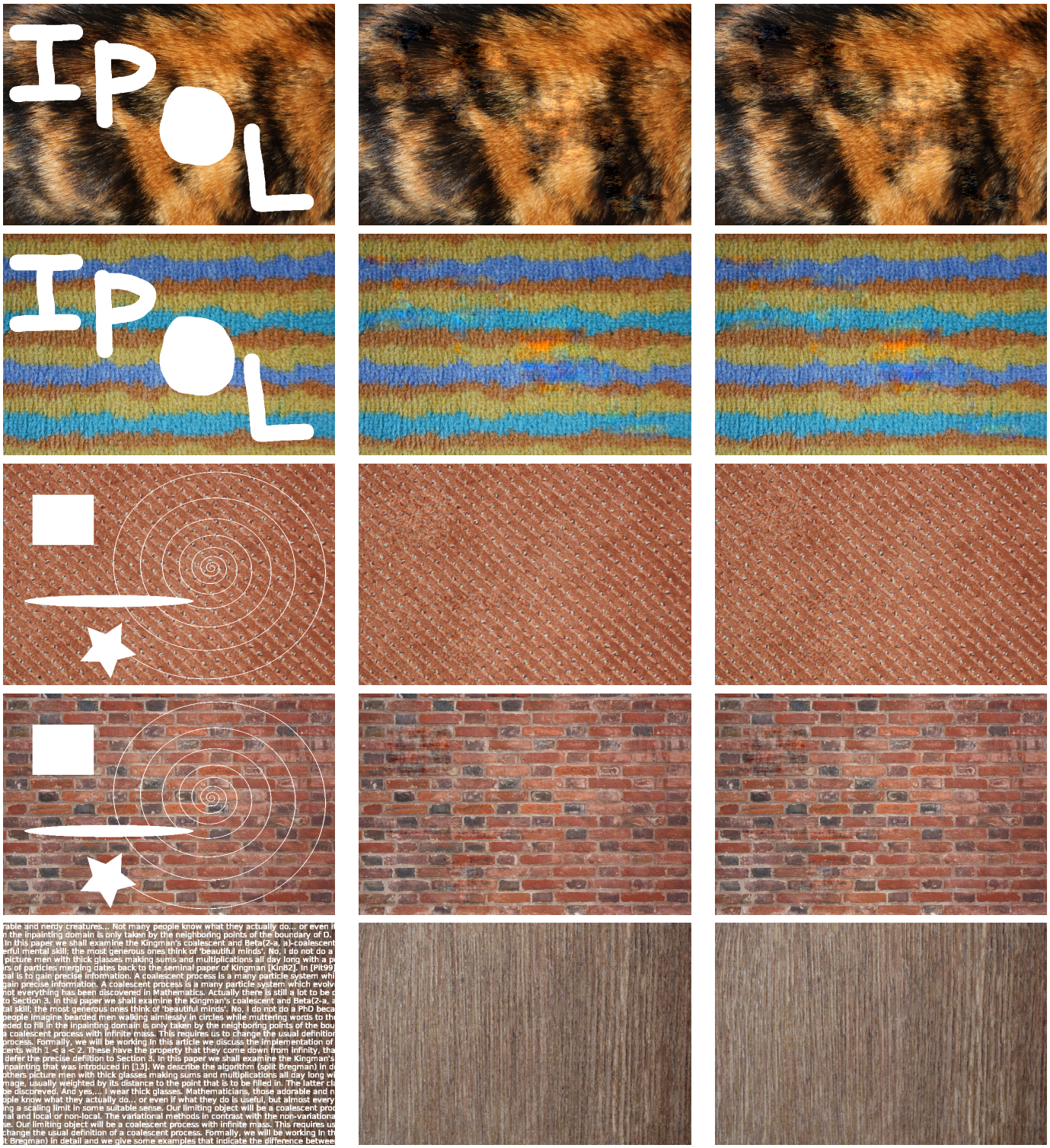
4.2 Discussion of the Thickness w of the Conditioning Set

Finally, we will discuss the parameter w which is the thickness of the conditioning border $\mathcal{C} = \partial_w M$. In Figure 6 we display several inpainting results obtained with different conditioning borders. Here again, there are only slight differences between these results and we suggest to flip between the different images for better comparison. In theory, increasing w allows to better grasp the textural behavior around the mask, but in practice, very good results are obtained with $w = 3$ or even $w = 1$. For the example in the first row of Figure 6 the result with $w = 3$ is slightly better than with $w = 1$ (the vertical correlations are slightly better extended through the mask). But one has to keep in mind that increasing w also increases the size of the linear system that has to be solved, and so one should increase $k_{\max}$ accordingly. We found that $w = 3$ is a good compromise to account for a sufficient neighborhood of the mask while keeping a reasonable number of iterations in the CGD.



Masked input | Results for $k_{\max} = 100$ | Results for $k_{\max} = 1000$

Figure 3: Examples of successful texture inpainting with $k_{\max} = 100$ and $k_{\max} = 1000$. All images have size 768×512. One observes that for $k_{\max} = 1000$ the color of the inpainted parts are better blended with the known area.



Masked input | Results for $k_{\max} = 100$ | Results for $k_{\max} = 1000$

Figure 4: Same as Figure 3 but with unsuccessful examples. The failures are mostly due to the fact that the Gaussian model is not accurate for the texture. For the last case, it is due to the irregularity of the mask that prevents a proper estimation of some characteristic frequency content for the wood texture.

Figure 5: Examples of texture inpainting with $k_{\max} = 100$ and $k_{\max} = 1000$. All images have size 128×128. Successful examples are presented in the left column while (relative) failure examples are presented in the right column (please zoom in for close inspection).

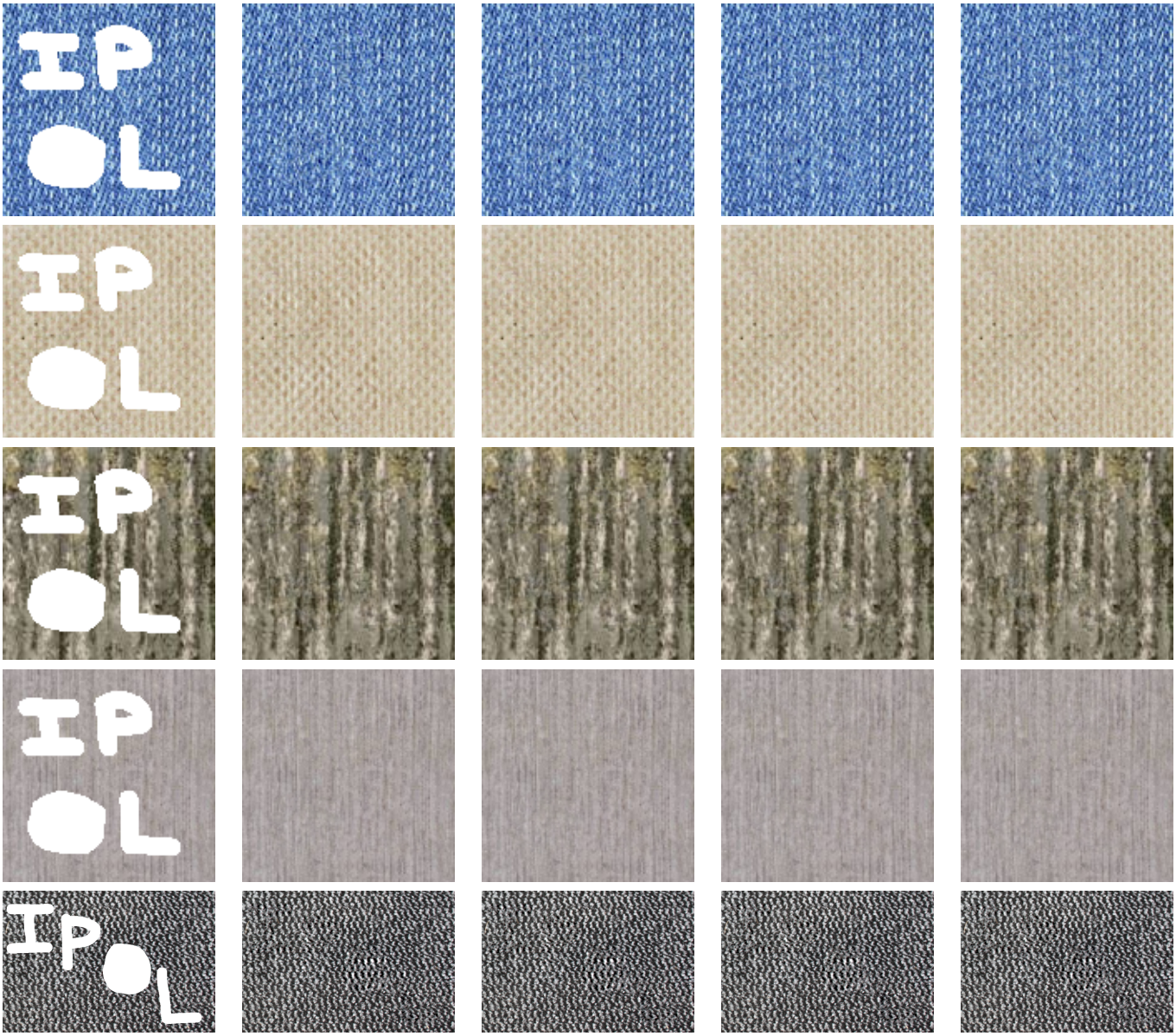


Figure 6: **Varying the thickness of the conditioning set.** The first column contains the masked exemplar. The other columns contain the inpainting result obtained with conditioning sets of thickness $w = 1, 3, 5, 10$.

Image Credits

All the textures shown in this article come from Bruno Galerne's personal collection (CC-BY).

References

[1] P. Arias, V. Caselles, and G. Facciolo, *Analysis of a Variational Framework for Exemplar-Based Image Inpainting*, Multiscale Modeling & Simulation, 10 (2012), pp. 473–514. https://doi.org/10.1137/110848281.

[2] P. Arias, G. Facciolo, V. Caselles, and G. Sapiro, *A variational framework for exemplar-based image inpainting*, International Journal of Computer Vision, 93 (2011), pp. 319–347. https://doi.org/10.1007/s11263-010-0418-7.

[3] J.F. Aujol, S. Ladjal, and S. Masnou, *Exemplar-based inpainting from a variational point of view*, SIAM Journal on Mathematical Analysis, 42 (2010), pp. 1246–1285. https://doi.org/10.1137/080743883.

[4] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, *Image Inpainting*, in Proceedings of SIGGRAPH, 2000, pp. 417–424. https://doi.org/10.1145/344779.344972.

[5] 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, 24 (2015), pp. 1809–1824. https://doi.org/10.1109/TIP.2015.2411437.

[6] T.F. Chan and J. Shen, *Mathematical models for local nontexture inpaintings*, SIAM Journal on Applied Mathematics, 62 (2002), pp. 1019–1043. https://doi.org/10.1137/S0036139900368844.

[7] A. A. Efros and T. K. Leung, *Texture synthesis by non-parametric sampling*, in Proceedings of International Conference on Computer Vision, vol. 2, 1999, pp. 1033–1038. https://doi.org/10.1109/ICCV.1999.790383.

[8] B. Galerne, Y. Gousseau, and J.-M. Morel, *Random Phase Textures: Theory and Synthesis*, IEEE Transactions on Image Processing, 20 (2011), pp. 257–267. https://doi.org/10.1109/TIP.2010.2052822.

[9] B. Galerne and A. Leclaire, *Texture Inpainting using Efficient Gaussian Conditional Simulation*, SIAM Journal on Imaging Sciences, 10 (2017), pp. 1446–1474. https://doi.org/10.1137/16M1109047.

[10] B. Galerne, A. Leclaire, and L. Moisan, *A Texton for Fast and Flexible Gaussian Texture Synthesis*, in Proceedings of EUSIPCO, 2014, pp. 1686–1690.

[11] ——, *Microtexture Inpainting Through Gaussian Conditional Simulation*, in Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2016.

[12] K. He and J. Sun, *Image completion approaches using the statistics of similar patches*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 36 (2014), pp. 2423–2435. https://doi.org/10.1109/TPAMI.2014.2330611.

[13] W. Kammerer and M. Nashed, *On the Convergence of the Conjugate Gradient Method for Singular Linear Operator Equations*, SIAM Journal on Numerical Analysis, 9 (1972), pp. 165–181. https://doi.org/10.1137/0709016.

[14] E. Levina and P.J. Bickel, *Texture synthesis and nonparametric resampling of random fields*, The Annals of Statistics, 34 (2006), pp. 1751–1773. https://doi.org/10.1214/009053606000000588.

[15] Y. Liu and V. Caselles, *Exemplar-based image inpainting using multiscale graph cuts*, IEEE Transactions on Image Processing, 22 (2013), pp. 1699–1711. https://doi.org/10.1109/TIP.2012.2218828.

[16] S. Masnou, *Disocclusion: a variational approach using level lines*, IEEE Transactions on Image Processing, 11 (2002), pp. 68–76. https://doi.org/10.1109/83.982815.

[17] S. Masnou and J.-M. Morel, *Level lines based disocclusion*, in Proceedings of IEEE International Conference on Image Processing, vol. 3, 1998, pp. 259–263. https://doi.org/10.1109/ICIP.1998.999016.

[18] A. Newson, A. Almansa, M. Fradet, Y. Gousseau, and P. Pérez, *Video Inpainting of Complex Scenes*, SIAM Journal on Imaging Sciences, 7 (2014), pp. 1993–2019. https://doi.org/10.1137/140954933.

[19] C.B. Schönlieb, *Partial Differential Equation Methods for Image Inpainting*, Cambridge University Press, 2015.

[20] L.Y. Wei and M. Levoy, *Fast texture synthesis using tree-structured vector quantization*, in Proceedings of SIGGRAPH, 2000, pp. 479–488. https://doi.org/10.1145/344779.345009.

[21] Y. Wexler, E. Shechtman, and M. Irani, *Space-Time Completion of Video*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29 (2007), pp. 463–476. https://doi.org/10.1109/TPAMI.2007.60.