IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2015–12–26.
Submitted on 2015–04–13, accepted on 2015–08–13.
ISSN 2105–1232 © 2015 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
http://dx.doi.org/10.5201/ipol.2015.136

Variational Framework for Non-Local Inpainting

Vadim Fedorov [1], Gabriele Facciolo [2], Pablo Arias [2]

[1] DTIC, Universitat Pompeu Fabra, Spain (vadim.fedorov@upf.edu)
[2] CMLA, ENS-Cachan, France ({facciolo, pablo.arias}@cmla.ens-cachan.fr)

Abstract

Image inpainting aims to obtain a visually plausible image interpolation in a region of the image in which data is missing due to damage or occlusion. Usually, the only available information is the portion of the image outside the inpainting domain. Besides its numerous applications, the inpainting problem is of theoretical interest since its analysis involves an understanding of the self-similarity present in natural images. In this work, we present a detailed description and implementation of three exemplar-based inpainting methods derived from the variational framework introduced by Arias et al.

Source Code

The implementation written in C++ for this algorithm is available in the IPOL web page of this article[1]. Compilation and usage instructions are included in the `README.md` file of the archive.

Keywords: image inpainting; variational methods; exemplar-based; nonlocal methods

1 Introduction

Image inpainting, also known as *image completion*, *disocclusion* or *object removal*, aims to obtain a visually plausible image interpolation in a region of the image in which data is missing due to damage or occlusion. Usually, to solve this problem, the only available data is the image outside the region to be inpainted. Inpainting has become a standard tool in digital photography for image retouching and video post-production. Besides its numerous applications, the problem is of theoretical interest since its analysis involves an understanding of the self-similarity present in natural images.

1.1 Context: Geometric and Texture Inpainting

Most inpainting methods found in the literature can be classified into two groups: *geometry-* and *texture-oriented* depending on how they characterize the redundancy of the image.

[1] http://dx.doi.org/10.5201/ipol.2015.136

Geometry-oriented methods. In this class of methods images are usually modeled as functions with some degree of smoothness, expressed for instance in terms of the curvature of the level lines or the total variation of the image. This smoothness assumption is exploited to interpolate the inpainting domain by continuing the geometric structures of the image (its level lines, or its edges), usually by means of a partial differential equation (PDE). Such PDEs can be derived from variational principles [31, 5, 14, 15, 20, 30], or inspired by phenomenological modeling [8, 11, 41]. These methods show good performance in propagating smooth level lines or gradients, but fail in the presence of texture. They are often referred to as *structure* or *cartoon* inpainting.

The geometry-oriented methods are said to be *local* in the sense that among all the data available from the image, they only use the information at the boundary of the inpainting domain.

Texture-oriented methods. Texture-oriented inpainting was born as an application of texture synthesis [19, 25]. Its recent development was triggered in part by the works of Efros and Leung [19], and Wei and Levoy [42] using non-parametric sampling techniques. In these works texture is modeled as a two dimensional probabilistic *graphical model*, in which the value of each pixel is conditioned by its neighborhood. These approaches rely directly on a sample of the desired texture to perform the synthesis. The value of each synthesized pixel x is copied from the center of a patch in the sample image, chosen to match the available portion of the patch centered at x.

This strategy has been extensively used for inpainting [9, 10, 16, 18, 19, 35]. As opposed to geometry-oriented inpainting, these so-called *exemplar-based* approaches are *non-local*. That is, to determine the value at x, the whole image may be traversed in the search for a matching patch. Thus the problem of exemplar-based inpainting can be stated [17] as that of finding a *correspondence map* $\varphi : O \to O^c$, which assigns to each point x in the inpainting domain O (a subset of the image domain $\Omega \subset \mathbb{R}^2$) a corresponding point $\varphi(x) \in O^c := \Omega \setminus O$ where the image is known (see Figure 1). The unknown part of the image is then synthesized using the map φ.

Early exemplar-based approaches to image inpainting [19, 42] use *greedy* procedures for computing the correspondence map. Their results are very sensitive to the order in which the pixels are processed [16, 35, 23]. To address this issue Demanet et al. [17] proposed to model the inpainting problem as an energy minimization in which the unknown is the correspondence map itself

$$\mathcal{E}(\varphi) = \int_O \int_{\Omega_p} |\hat{u}(\varphi(x+h)) - \hat{u}(\varphi(x)+h)|^2 dh dx, \tag{1}$$

where $\hat{u} : O^c \to \mathbb{R}$ is the known part of the image, and Ω_p is the patch domain (a neighborhood of the origin $0 \in \mathbb{R}^2$). The unknown part of the image is then computed as $u(x) = \hat{u}(\varphi(x))$, $x \in O$. Thus φ should map a pixel x and its neighborhood in such a way that the resulting patch is close to the one centered at $\varphi(x)$. The energy (1) is highly non-convex and no effective way to minimize it is known [3]. Hence, other authors have proposed simpler optimization problems for determining such correspondence map.

For instance, in [27] the correspondence map estimation is formulated as a probabilistic inference on a graphical model defined on a coarse image lattice. A message passing algorithm could then be applied to compute a correspondence map on this coarse grid. Other authors [38, 29] propose to minimize the energy (1) for the smallest possible patch domain Ω_p: the center of the patch plus its four immediate neighbors in the discrete image rectangular lattice. The minimization is seen as a graph labeling problem on the lattice, where the labels are a discrete set of shifts t (or equivalently, the positions on the known portion of the image). In this case, the energy (1) corresponds to a pairwise Markov Random Field and is minimized using graph cuts.

Another approach [26, 43, 28] is to treat the unknown image as an auxiliary variable. The

Figure 1: **Inpainting problem.** Left: on a rectangular image domain Ω, missing data u in a region O has to be reconstructed using the available image $\hat{u}$ over $O^c := \Omega \setminus O$. A patch centered at $x \in O$ is denoted by $p_u(x)$. The set of centers of incomplete patches is $\widetilde{O} := O + \Omega_p$, where Ω_p denotes the patch domain. Middle: the image shows a completion obtained using the *patch NL-medians*, a scheme derived from the general formulation presented in this work. Right: the resulting completion shows a correspondence map which is a piece-wise translation. The red curves show the boundaries between the regions of constant translation. In each of these regions, the image is copied rigidly from a corresponding region in O^c.

resulting energy functional can be regarded as a relaxation of (1)

$$\mathcal{E}(u, \varphi) = \int_{\widetilde{O}} \int_{\Omega_p} |u(x+h) - \hat{u}(\varphi(x)+h)|^2 dh dx, \tag{2}$$

where $\widetilde{O} := O + \Omega_p$ denotes the set of centers of patches that intersect the inpainting domain O (see Figure 1). The energy is optimized using an alternating scheme with respect to the variables u and φ. The unknown image is determined as part of the optimization process, instead of being constrained to be $u(x) = \hat{u}(\varphi(x))$. Although this relaxation is still non-convex, the alternating minimization scheme converges to a critical point of the energy. This approach has been applied to video denoising by considering 3D patches with impressive results [43, 34].

In [2] a formulation is proposed which can be seen as a generalization of (2) in two different aspects:

1. By considering different patch similarity criteria other than the squared L^2-norm. Via the selection of the patch similarity criterion different inpainting schemes can be naturally derived. Four schemes are discussed in [2], *patch NL-means*, *-medians*, *-Poisson*, and *gradient medians* corresponding to similarity criteria based on L^2- and L^1-norms between patches or their gradients. The *patch NL-means* is related to the inpainting methods of Kawai et al. [26] and Wexler and Irani [43]. Methods related to the *patch NL-Poisson* and *patch NL-medians* have been used by Kwatra et al. [28] in the context of texture synthesis.

2. By encoding self-similarity as a non-local weight function $w : \widetilde{O} \times \widetilde{O}^c \to \mathbb{R}$, which serves as a probabilistic correspondence. The resulting weights w correspond to the exponential weights used for non-local denoising and regularization [4, 12]. A parameter T controls the spread of the similarity weights. In particular, when $T = 0$ the similarity weights "converge" to a correspondence map $\varphi : \widetilde{O} \to \widetilde{O}^c$. This can be made rigorous through the notion of Gamma convergence (Section 5 of [1]). An analysis of these models can be found in [1].

The general framework of [2] can be applied not only for inpainting but as regularizer in other inverse problems in which the similarity weights are unknown and have to be estimated together with

the image ([21], [37], [36]). For the inpainting application it is best suited to take $T = 0$, because $T > 0$ implicates also denoising within the inpainting domain. Therefore, in this work we focus on the case $T = 0$.

The inpainting is performed by an iterative minimization process, alternating between image and correspondence map updates. This iterative process tends to generate a sort of patchwork of segments copied from the known portion of the image. An example is shown in Figure 1. The transition between copied segments takes place at a narrow band along the boundary between them. The four inpainting schemes differ in the way this transition is done (and in the partition found). Methods based on the L^2-norm perform a smooth blending, whereas those based on the L^1-norm favor sharper transitions.

In particular, *patch NL-Poisson* and *patch NL-gradient medians* combine the exemplar-based interpolation with PDE-based diffusion schemes. This results in a smoother continuation of the information across the boundary and inside the inpainting domain, and in a better propagation of structures. Furthermore, the inclusion of gradients in the patch similarity criterion allows to handle additive brightness changes.

1.2 Contributions

This publication describes the implementation of the inpainting framework [2] for the case $T = 0$, providing a detailed description and C++ source code for three of the inpainting schemes derived from it: *patch NL-means*, *patch NL-medians* and *patch NL-Poisson*. The code was designed so that other inpainting schemes, derived from the same framework with different patch similarity criteria, can be easily added (such as the *patch NL-gradient medians*).

Although the code is written with more emphasis on readability than on performance, for the computation of the correspondence map we use a parallel version of the PatchMatch Algorithm [6], an efficient algorithm to approximate optimal correspondences.

After introducing the notation in the next subsection, in Section 2 we introduce the continuous variational framework [2] and the inpainting energies derived from it. Section 3 focuses on the discrete multiscale minimization of these energies. The efficient computation of the patch correspondences is done using the PatchMatch Algorithm [6], which is described in Section 3.3. Section 4 presents some experiments and Section 2.3 some extensions of the framework. The organization of the code is commented in Appendix A.

1.3 Notation

Images are denoted as functions $u : \Omega \to \mathbb{R}$, where Ω denotes the image domain, a rectangle in $\mathbb{R}^2$. We commonly refer to points in Ω as pixels and denote them by x, $\hat{x}$, z, $\hat{z}$. Positions inside the patch are denoted as y. A patch of u centered at x is denoted by $p_u(x) := p_u(x, \cdot) : \Omega_p \to \mathbb{R}$, where Ω_p is a rectangle centered at 0. The patch is defined by $p_u(x, y) := u(x + y)$, with $y \in \Omega_p$. Let $O \subset \Omega$ be the hole or inpainting domain, and $O^c = \Omega \setminus O$. We assume that O is an open set with Lipschitz boundary. We still denote by $\hat{u} : O^c \to \mathbb{R}$ the known part of the image u: $\hat{u} := u|_{O^c}$.

We denote by $\widetilde{\Omega}$ the set of centers of patches contained in the image domain, i.e. $\widetilde{\Omega} = \{x \in \Omega : x + \Omega_p \subseteq \Omega\}$. As was defined in the Introduction, we take $\widetilde{O}$ as the set of centers of patches that intersect the hole, i.e. $\widetilde{O} := O + \Omega_p = \{x \in \Omega : (x + \Omega_p) \cap O \neq \emptyset\}$. For a simplified presentation, we assume that $\widetilde{O} \subseteq \widetilde{\Omega}$, i.e. every pixel in $\widetilde{O}$ is the center of a patch contained in Ω. We denote $\widetilde{O}^c = \widetilde{\Omega} \setminus \widetilde{O}$. Thus, patches $p_{\hat{u}}(y)$ centered at points $y \in \widetilde{O}^c$ are contained in O^c (see Figure 1). Further notation will be introduced in the text.

2 Variational Framework

Let us briefly review the inpainting framework proposed in [2] (we will focus here on the particular case when $T \to 0$, please refer to [2] for a more general presentation).

The image inpainting problem is posed as the minimization of the following energy

$$\mathcal{E}_E(u, \varphi) = \int_{\widetilde{O}} E(p_u(x) - p_{\hat{u}}(\varphi(x)))dx, \tag{3}$$

where E is a *patch error function* which measures patch similarity. Similarly to Equation (2), this energy forces every patch inside the inpainting domain O to be similar to some patch in $\tilde{O}^c$. Different choices of E yield different methods.

2.1 The Patch Error Function E

We consider [2] patch error functions $E : \Omega_p \to \mathbb{R}^+$ defined either as the weighted sum of pixel errors

$$E(p_u(x) - p_{\hat{u}}(\hat{x})) := g * \mathrm{e}(u(x + \cdot) - \hat{u}(\hat{x} + \cdot)) = \int_{\Omega_p} g(h)\mathrm{e}(u(x + h) - \hat{u}(\hat{x} + h))dh,$$

where $\mathrm{e} : \mathbb{R} \to \mathbb{R}^+$, or gradient errors

$$E(p_u(x) - p_{\hat{u}}(\hat{x})) := g * \mathrm{e}(\nabla u(x + \cdot) - \nabla\hat{u}(\hat{x} + \cdot)) = \int_{\Omega_p} g(h)\mathrm{e}(\nabla u(x + h) - \nabla\hat{u}(\hat{x} + h))dh.$$

where $\mathrm{e} : \mathbb{R}^2 \to \mathbb{R}^+$. Here, $g : \mathbb{R}^2 \to \mathbb{R}^+$ denotes a suitable *intra-patch* kernel function, a nonnegative function such that $\int g(h)\, dh = 1$. For example the Gaussian weights $g(h) = \frac{1}{Z}\exp\left(-\frac{\|h\|^2}{2a^2}\right)$, with standard deviation a and normalization factor Z. We will consider three concrete patch error functions.

Patch non-local means. In this case we use $\mathrm{e}(r) = |r|^2$ and the patch error function is a weighted squared L^2-norm that we denote by

$$E_2(p_u(x) - p_{\hat{u}}(\hat{x})) = \|p_u(x) - p_{\hat{u}}(\hat{x})\|^2_{g,2} = g * |u(x + \cdot) - \hat{u}(\hat{x} + \cdot)|^2. \tag{4}$$

Patch non-local medians. If we set $\mathrm{e}(r) = |r|$ then the patch error function results in a weighted L^1-norm

$$E_1(p_u(x) - p_{\hat{u}}(\hat{x})) = \|p_u(x) - p_{\hat{u}}(\hat{x})\|_{g,1} = g * |u(x + \cdot) - \hat{u}(\hat{x} + \cdot)|. \tag{5}$$

Patch non-local Poisson. Taking a pixel error $\mathrm{e}(\,\cdot\,)$ that is a convex combination (with parameter $\lambda \in [0, 1)$) of intensity and gradient errors results in the following patch error function:

$$E_{\lambda,2}(p_u(x) - p_{\hat{u}}(\hat{x})) = \lambda\|p_u(x) - p_{\hat{u}}(\hat{x})\|^2_{g,2} + (1 - \lambda)\|p_u(x) - p_{\hat{u}}(\hat{x})\|^2_{g,2,\nabla} = g * \left(\lambda|u(x + \cdot) - \hat{u}(\hat{x} + \cdot)|^2 + (1 - \lambda)|\nabla u(x + \cdot) - \nabla\hat{u}(\hat{x} + \cdot)|^2\right). \tag{6}$$

Note that if we set $\lambda = 1$ we get E_2.

By plugging each of these patch error functions in the energy $\mathcal{E}_E$ we get different inpainting functionals. As it will be discussed below, the patch error function determines not only the similarity criterion but also the image synthesis, thus it is a key element in the framework.

Additional patch error functions could be considered. For example in [2] the L_1 norm between patches of the gradient is also considered.

2.2 Minimization of the Energies

The objective $\mathcal{E}_E$ in (3) is non-convex, and we can only compute a local minimum. To that aim, we use an alternating minimization algorithm. At each iteration, two optimization steps are solved: the constrained minimization of $\mathcal{E}_E$ with respect to φ while keeping u fixed; and the minimization of $\mathcal{E}_E$ with respect to u with φ fixed. This procedure is detailed next and summarized in Algorithm 1.

Algorithm 1: Alternate minimization of $\mathcal{E}_E(u, \varphi)$.

input : Initial condition $u_0(x)$ with $x \in O$, tolerance $\tau > 0$
output: Inpainted image u_{k+1}, offset map φ_k
repeat
  $\varphi_k \leftarrow \arg\min_{\varphi} \mathcal{E}_E(u_k, \varphi)$. //§ 2.2.1: Correspondence update
  $u_{k+1} \leftarrow \begin{cases} \arg\min_u \mathcal{E}_E(u, \varphi_k) \\ \text{subject to } u_{k+1}(x) = \hat{u}(x),\ \forall x \in O^c \end{cases}$ //§ 2.2.2: Image update: (11), (12), or (19)
until $\|u_{k+1} - u_k\| < \tau$

2.2.1 Correspondence Update Step

When u is fixed, the minimization with respect to φ can be done independently for each $x \in \widetilde{O}$. This amounts to determine the location in $\widetilde{O}^c$ of the most similar patch to $p_u(x)$, which is said to be the *nearest neighbor* of $p_u(x)$:

$$\varphi(x) \in \arg\min_{\hat{x} \in \widetilde{O}^c} E(p_u(x) - p_{\hat{u}}(\hat{x})). \tag{7}$$

In Section 3.3 we describe an efficient way to compute the nearest neighbor for all the pixels in $\widetilde{O}$ without using an exhaustive search.

2.2.2 Image Update Step

Before moving to the derivation of the image update step for the different error functions: (4), (5), and (6), let us remark that with the change of variables $z := x + h$, $\hat{z} := \hat{x} + h$, the energy (3) can be expressed as an accumulation of pixel errors

$$\mathcal{E}_E(u) = \int_{\widetilde{O}} \int_{\Omega_p} g(h) \mathrm{e}(u(x+h) - \hat{u}(\varphi(x) + h)) dh dx \quad = \int_O \int_{O^c} m(z, \hat{z}) \mathrm{e}(u(z) - \hat{u}(\hat{z})) d\hat{z} dz + \mathrm{C}, \tag{8}$$

where C is a constant term, and the *pixel influence weights* $m : O \times O^c \to \mathbb{R}^+$ is defined as

$$m(z, \hat{z}) = \int_{\Omega_p} g(h) \chi_{\widetilde{O}}(z - h) \delta_{\varphi(z-h)}(\hat{z} - h) dh = \int_{\Omega_p} g(h) \chi_{\widetilde{O}}(z - h) \delta_{\varphi(z-h)+h}(\hat{z}) dh. \tag{9}$$

The characteristic function $\chi_{\widetilde{O}}(x)$ (1 inside $\widetilde{O}$ and 0 otherwise) zeroes out the terms for which $z - h$ falls out of $\widetilde{O}$, where φ is not defined.

For each pair of pixels $(z, \hat{z}) \in O \times O^c$, $m(z, \hat{z})$ weights the effective contribution of the pixel error between $u(z)$ and $\hat{u}(\hat{z})$ in the total value of the energy. The quantity $m(z, \hat{z})$ is determined by integrating the correspondences between all patches that overlap $\hat{z}$ and those that overlap z in the *same relative position* (shown in Figure 2).

Let us also introduce the following notation, for $z \in O$:

$$k(z) := \int_{O^c} m(z, \hat{z}) d\hat{z}, \tag{10}$$

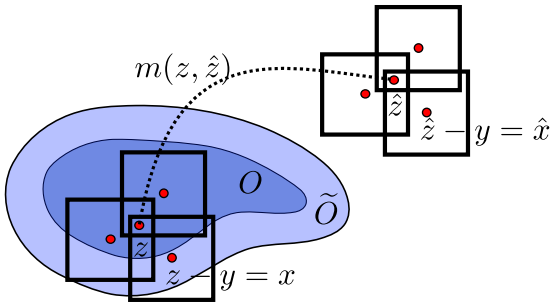


Figure 2: **Patch non-local inpainting.** The value at $z \in O$ is computed using contributions from all the patches that overlap it, i.e. those centered at $x \in \widetilde{O}$ such that $z = x + h$ with $h \in \Omega_p$. The influence function $m(z, \hat{z})$ accounts for the contributions from patches centered at $\hat{z} - h$ to $z - h$.

Since g is normalized, we have that $k(z) = 1$ for all $z \in O$. Thus, $m(z, \cdot)$ can be interpreted as a probability density function. Later on, we will consider *confidence weights* for which $k(z) \neq 1$. For this reason we will keep the notation $k(z)$ in the following derivations.

Patch non-local means. Taking the weighted squared L^2-norm $\|p_u(x) - p_{\hat{u}}(\hat{x})\|^2_{g,2}$ as a patch error function (4), the energy (8) becomes quadratic on u. Thus its minimum for a fixed correspondence φ can be computed explicitly as a non-local average of the known pixels:

$$u(z) = \frac{1}{k(z)} \int_{O^c} m(z, \hat{z}) \hat{u}(\hat{z}) \mathrm{d}\hat{z}, \tag{11}$$

for $z \in O$. By expanding m we note that each patch overlapping z contributes a value for the current pixel z

$$u(z) = \frac{1}{k(z)} \int_{\Omega_p} g(h) \hat{u}(\varphi(z - h) + h) dh.$$

Patch non-local medians. Considering the L^1-norm patch error function (5), corresponds to taking $\mathrm{e}(x) = |x|$ in (8). The Euler equation for u can be formally written as

$$[\delta_u \mathcal{E}_E(u)](z) = \int_{O^c} \mathrm{sign}[u(z) - \hat{u}(\hat{z})] m(z, \hat{z}) d\hat{z} \ni 0. \tag{12}$$

This expression is multivalued, since $\mathrm{sign}(r) := r/|r|$ if $|r| > 0$ and $\mathrm{sign}(r) \in [-1, 1]$ if $r = 0$. Its solution $u(z)$ for every $z \in O$ is obtained as a weighted median of the pixels of the complement O^c, with weights $m(z, \cdot)$.

Both schemes presented so far perform inpainting by transferring (averages or medians) known gray levels into the inpainting domain. As we will see next, using a patch error function based on the gradient of the image yields a method that transfers gradients and computes the resulting image as the solution of a PDE. This results in better continuation properties of the solution, in particular at the boundary of the inpainting domain.

Patch non-local Poisson. The patch NL-Poisson combines weighted L^2-norms of intensity (5) and gradients (6). The resulting energy is

$$\mathcal{E}_{\lambda,2}(u) = \int_O \int_{O^c} m(z, \hat{z}) [(1 - \lambda) |\nabla u(z) - \nabla \hat{u}(\hat{z})|^2 + \lambda (u(z) - \hat{u}(\hat{z}))^2] d\hat{z} dz. \tag{13}$$

Note that (13) can be rewritten as

$$\mathcal{E}_{\lambda,2}(u) \propto \int_O k(z)|\nabla u(z) - \boldsymbol{v}(z)|^2 dz + \frac{\lambda}{1-\lambda}\int_O k(z)(u(z) - f(z))^2 dz + C, \tag{14}$$

where C is a constant term,

$$f(z) := \frac{1}{k(z)}\int_{O^c} m(z,\hat{z})\hat{u}(\hat{z})d\hat{z} \tag{15}$$

is the solution of the patch NL-means image update step, and the field $\boldsymbol{v} : O \to \mathbb{R}^N$ is given by

$$\boldsymbol{v}(z) := \frac{1}{k(z)}\int_{O^c} m(z,\hat{z})\nabla\hat{u}(\hat{z})d\hat{z}. \tag{16}$$

This energy balances two terms. The first one imposes the gradients of u to be close (in the L^2 sense) to a *guiding* vector field $\boldsymbol{v}$ computed as a non-local weighted average of the image gradients in the known portion of the image. The second term corresponds to a quadratic attachment to the solution of the patch NL-means image update.

In this case the Euler-Lagrange equation w.r.t. u is the *screened Poisson equation*

$$\begin{cases} \operatorname{div}[k(z)\nabla u(z)] - \dfrac{\lambda}{(1-\lambda)}k(z)u(z) = & \\ \quad \operatorname{div}[k(z)\boldsymbol{v}(z)] - \dfrac{\lambda}{(1-\lambda)}k(z)f(z), & z \in O, \\ u(z) = \hat{u}(z), & z \in \partial O \setminus \partial\Omega, \\ \nabla u(z)\cdot n_\Omega(z) = 0, & z \in \partial O \cap \partial\Omega. \end{cases} \tag{17}$$

Here $n_\Omega(z) \in \mathbb{R}^N$ denotes the outgoing normal to Ω, for $z \in \partial\Omega$. The problem is linear and in a discrete setting it can be solved for instance with a conjugate gradient algorithm.

When $\lambda = 0$ the attachment to f vanishes and only gradients are transfered to the inpainting domain, the resulting PDE is a Poisson equation. In this case, the patch similarity (that determines m) is computed based solely on the gradients. However, the gradient is usually not a good feature for measuring the patch similarity, and it is convenient to consider also the gray level/color data. Typically we will set $\lambda \leqslant 0.1$, in this way some intensity information is included in the computation of the correspondences, without departing too much from the Poisson equation.

2.3 Extension: Decoupling Image and Correspondence Update Steps

In the variational setting, the image synthesis and the correspondence update are *coupled* through a unique patch error function E. One can envision a broader family of (*non-variational*) methods in which different patch comparison criterions can be chosen independently for the correspondence update step and for the image update step. If we denote by E_u and E_φ the patch error functions for image update and the correspondence update respectively, we have the following algorithm (Algorithm 2).

This is useful in practice for gradient-based methods. Gradient-based methods produce smooth interpolations and enforce the continuity of the image at the boundary of the inpainting domain, which are generally desirable features. For this to be done variationally the correspondences φ have to be computed using patches of the gradient, which in most cases do not provide a reliable measure of patch similarity. For this reason for patch NL-Poisson we define the patch error function as a convex combination between a gradient patch error function and the squared L^2-norm of intensities (the patch error function of patch NL-means). In this way, by controlling the coefficient λ of the convex

Algorithm 2: Alternate minimization of $\mathcal{E}_{E_u}$ and $\mathcal{E}_{E_\varphi}$.

Initialization: choose u^0 with $\|u^0\|_\infty \leq \|\hat{u}\|_\infty$.
for $k \in \mathbb{N}$ **do**
  $\varphi_{k+1} \leftarrow \arg\min_{\varphi} \mathcal{E}_{E_\varphi}(u_k, \varphi)$;
  $u_{k+1} \leftarrow \arg\min_{u} \mathcal{E}_{E_u}(u, \varphi_{k+1})$;

combination we can take the image values into account for the computation of the correspondences, and at the same time, to synthesize the image with a diffusion PDE.

However, in many cases, one cannot find a value of λ which balances a gradient-based blending of the image and at the same time produces reliable correspondences. For such cases it is useful to extend gradient-based methods by considering two mixture coefficients: $\lambda_\varphi \in [0, 1]$ for the correspondences update step and $\lambda_u \in [0, 1]$ controlling the image synthesis. This allows to combine the benefits of intensity- and gradient-based methods into a single algorithm.

3 Implementation

In this section we describe the main implementation aspects of the schemes presented in the previous section. Section 3.2 details the implementation of the different image update steps. Section 3.3 is devoted to the update of the correspondence map (or nearest neighbor field), for what we use the PatchMatch Algorithm [6], which permits to efficiently compute an approximate nearest neighbor field. Finally, in Section 3.4 we describe a multiscale scheme, necessary to avoid bad local minima.

3.1 Discrete Setting and Notation

Throughout this section we consider discrete images defined over a rectangular bounded domain $\Omega := \{0, \dots, N\}^2$ and corresponding discrete versions of the inpainting domain $\widetilde{O}$ and its complement $\widetilde{O}^c$. To avoid a cumbersome notation, we slightly modify it in this section (for instance some arguments of functions will be denoted as subindices). The discrete energy reads

$$\mathcal{E}_E(u, \varphi) = \sum_{x \in \widetilde{O}} E(p_u(x) - p_{\hat{u}}(\varphi(x))). \tag{18}$$

Patches are now functions defined on a discrete domain Ω_p. The patch error function E is defined as the weighted sum of pixel errors

$$E(p_u(x) - p_{\hat{u}}(\hat{x})) = \sum_{h \in \Omega_p} g_h \mathrm{e}(u_{x+h} - \hat{u}_{\hat{x}+h}),$$

where $\mathrm{e} : \mathbb{R} \to \mathbb{R}^+$, or gradient errors

$$E(p_u(x) - p_{\hat{u}}(\hat{x})) = \sum_{h \in \Omega_p} g_h \mathrm{e}(\nabla u_{x+h} - \nabla \hat{u}_{y+h}),$$

where $\mathrm{e} : \mathbb{R}^2 \to \mathbb{R}^+$. Here, $g : \mathbb{Z}^2 \to \mathbb{R}^+$ denotes the *intra-patch* kernel function, which is nonnegative, with support Ω_p and $\sum_{h \in \mathbb{Z}^2} g_h = 1$.

We need to define the discrete version of the gradient and divergence operators. We will use the notation $[A \to B] = \{f : A \to B\}$, the set of functions from A to B.

We define $\nabla : [\Omega \to \mathbb{R}] \to [\Omega \to \mathbb{R}^2]$ as

$$\nabla u^1_{i,j} := \begin{cases} 0 & \text{if } i = N, \\ u_{i+1,j} - u_{i,j} & \text{if } i < N, \end{cases}$$

and similarly for the component $\nabla u^2_{i,j}$. The notation $z = (i, j)$ refers to locations on the image. Let us also define a divergence operator $\nabla\cdot : [\Omega \to \mathbb{R}^2] \to [\Omega \to \mathbb{R}]$,

$$\nabla \cdot p_{i,j} := \begin{cases} p^1_{i,j} & \text{if } i = 0, \\ -p^1_{i-1,j} & \text{if } i = N, \\ p^1_{i,j} - p^1_{i-1,j} & \text{otherwise,} \end{cases} + \begin{cases} p^2_{i,j} & \text{if } j = 0, \\ -p^2_{i,j-1} & \text{if } j = N, \\ p^2_{i,j} - p^2_{i,j-1} & \text{otherwise.} \end{cases}$$

These operators incorporate Neumann boundary conditions on the boundary of Ω, and are dual operators, i.e. denoting $\langle\cdot,\cdot\rangle$ the usual scalar products in $[\Omega \to \mathbb{R}]$ and $[\Omega \to \mathbb{R}^2]$, $\langle \nabla u, p\rangle = -\langle u, \nabla\cdot p\rangle$, for all $u \in [\Omega \to \mathbb{R}]$, $p \in [\Omega \to \mathbb{R}^2]$.

3.2 Image Update Step

For the computation of the image update step, we will rewrite the energy using the pixel influence weights $m : O_e \times O_e^c \to \mathbb{R}^+$

$$m_{z\hat{z}} = \sum_{h \in \Omega_p} g_h \delta_{\varphi(z-h)}(\hat{z} - h)\chi_{\widetilde{O}}(\hat{z} - h).$$

Correspondingly we also define the normalization factor $k_z := \sum_{\hat{z}\in O^c} m_{z\hat{z}}$.

Patch NL-means and patch NL-Poisson. The discrete version of the patch NL-Poisson energy is given by

$$\mathcal{E}_{\lambda,2}(u) = \sum_{z\in O_e} \sum_{\hat{z}\in O_e^c} m_{z\hat{z}}[(1-\lambda)|\nabla u_z - \nabla \hat{u}_{\hat{z}}|^2 + \lambda(u_z - \hat{u}_{\hat{z}})^2].$$

The energy is quadratic in u and can be minimized by solving the following linear equation:

$$\begin{cases} (1-\lambda)\nabla\cdot[k_z\nabla u_z] - \lambda k_z u_z = (1-\lambda)\nabla\cdot[k_z \boldsymbol{v}_z] - \lambda k_z f_z, & z \in O_e, \\ u(z) = \hat{u}(z), & z \in O_e \setminus O, \end{cases} \tag{19}$$

where we have defined

$$f_z := \frac{1}{k_z} \sum_{\hat{z}\in O_e^c} m_{z\hat{z}} \hat{u}_{\hat{z}} \qquad \text{and} \qquad \boldsymbol{v}_z := \frac{1}{k_z} \sum_{\hat{z}\in O_e^c} m_{z\hat{z}} \nabla \hat{u}_{\hat{z}}.$$

Equation (19) can be solved efficiently with a conjugate gradient Algorithm [40].

When $\lambda = 1$, we recover the NL-means case and the solution can be computed directly as $u_z = f_z$.

Patch NL-medians. The discrete version of the patch NL-medians energy is given by

$$\mathcal{E}_1(u) = \sum_{z\in O} \sum_{\hat{z}\in O^c} m_{z\hat{z}} |u_z - \hat{u}_{\hat{z}}|.$$

This problem can be solved for each z independently, and its solution is given by a weighted median (with weights $m_{z\hat{z}}$) of the known values $\hat{u}_{\hat{z}}$. To compute the weighted median the values $\hat{u}_{\hat{z}}$ are first sorted, then the vector is traversed while accumulating the weights, the median corresponds to the element for which the accumulator attains half of the total weight.

Algorithm 3: PatchMatch

input : image u, inpainting domain $\widetilde{O}$, target image $\hat{u}$, I iterations
output: offset field θ *// can be converted to correspondences:* $\varphi_x = x + \theta_x$

$\theta \longleftarrow$ Randomly initialize the entire offset map
(x_n), $n = 1, \dots, |\widetilde{O}| \longleftarrow$ Raster ordering of the pixels in $\widetilde{O}$

for iter $= 0, \dots, I-1$ **do**
  for $n = 0, \dots, |\widetilde{O}| - 1$ **do**

    // Offset propagation
    if (iter mod 2) $= 0$ **then**
      $x = (i, j) \longleftarrow x_n$
      $\mathbb{T} \longleftarrow \{\theta_{i,j}\,,\, \theta_{i,j-1}\,,\, \theta_{i-1,j}\}$ *// candidate offsets for raster (up and left)*
    else
      $x = (i, j) \longleftarrow x_{|\widetilde{O}|-n}$
      $\mathbb{T} \longleftarrow \{\theta_{i,j}\,,\, \theta_{i,j+1}\,,\, \theta_{i+1,j}\}$ *// candidate offsets for inverse raster (down and right)*
    $\theta_x \longleftarrow \underset{t \in \mathbb{T}}{\arg\min}\, E(p_u(x) - p_{\hat{u}}(x+t))$ *// keep the offset yielding the minimum error*

    // Random search
    $\mathbb{S} \longleftarrow$ Generate set of random offsets around θ_x *// according to (21)*
    $\theta_x \longleftarrow \underset{t \in \{\{\theta_x\} \cup \mathbb{S}\}}{\arg\min}\, E(p_u(x) - p_{\hat{u}}(x+t))$

3.3 PatchMatch for Computing the Nearest Neighbor Field

The updating of correspondences is the most time consuming step in the minimization of the inpainting energies (Algorithm 1). A brute force search for correspondences requires $\mathcal{O}(|\widetilde{O}||\widetilde{O}^c||\Omega_p|)$ operations. In this section we describe (see Algorithm 3) the PatchMatch algorithm [6], which efficiently finds an approximate solution to this problem. Following [6], let us express the correspondence map as *offsets* $\theta_x = \varphi_x - x \in \mathbb{Z}^2$. We need to solve the following problem:

$$\theta_x \in \underset{t \in \mathbb{Z}^2 \,:\, x+t \in \widetilde{O}_e^c}{\arg\min}\; E(p_u(x) - p_{\hat{u}}(x+t)), \quad \text{for all } x \in \widetilde{O}_e. \tag{20}$$

Although the patch error does not have to be a metric, we refer to $p_{\hat{u}}(x+\theta_x)$ as the *nearest patch* or *nearest neighbor* of $p_u(x)$, and to the offset map $\theta : \widetilde{O} \to \widetilde{O}^c$ as the *nearest neighbor field* (NNF).

The search for the nearest neighbor is performed simultaneously over all the points in $\widetilde{O}$ based on the following heuristic: since query patches overlap, the offset θ_x of a good match at x is likely to be a good match for points close to x.

PatchMatch is an iterative algorithm which starts from a random initialization, when each pixel x in the extended inpainting domain $\widetilde{O}$ is associated with a candidate correspondence $\varphi_x \in \widetilde{O}^c$ following a uniform distribution over $\widetilde{O}^c$. The corresponding offset is then $\theta_x = \varphi_x - x$. After the initialization, each iteration goes through all pixels in $\widetilde{O}$ and performs two operations at each pixel: offsets propagation and random search. In odd iterations, pixels are traversed in raster order (from left to right, top to bottom), while in even iterations the inverse ordering is used.

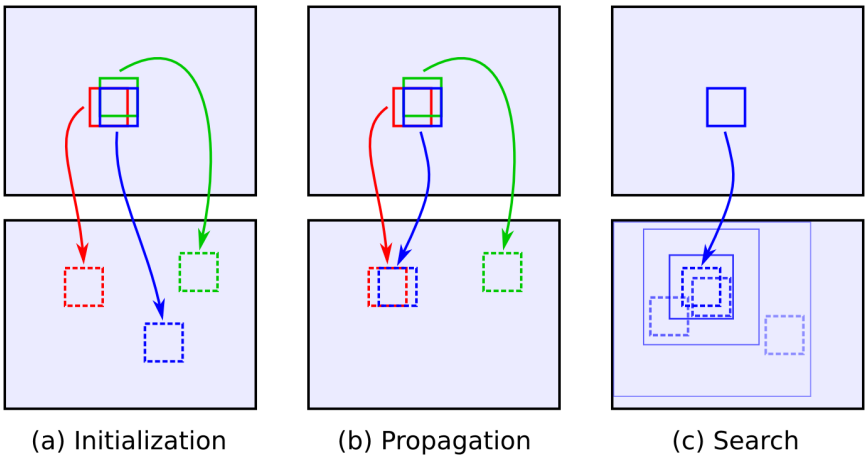


Figure 3: Steps of the PatchMatch algorithm: a) assign offsets at random; b) check left and top neighbors of the blue patch for the better offset (suppose iteration is odd); c) search for the better offsets at random in sequentially decreasing windows.

Offsets propagation. Let us consider an odd iteration, when pixels are traversed in raster order. At the pixel $x = (i, j)$, the offsets of its left and top neighbors are tested as candidate offsets for x. Among $\theta_{i,j}, \theta_{i-1,j}$ and $\theta_{i,j-1}$, the offset yielding the least patch error is kept as candidate. Note that the left and top neighbors have already been visited in the propagation scan. In a backward propagation step the pixels are traversed in the inverse order. Correspondingly, when visiting a pixel, we consider the offsets from its right and bottom neighbors.

Random search. The result of the propagation is refined with a random search that samples a set of candidate offsets for the pixel x. The patch error of each one of these candidate offsets is compared with the current offset θ_x, and the one with minimum error is kept. Each sample $t_i \in \mathbb{Z}^2$, $i = 0, 1, \ldots$ is drawn from a uniform distribution in a square neighborhood centered at the current offset, i.e.

$$t_i \sim U([-\alpha^i N_{\max}, \alpha^i N_{\max}]^2) + \theta_x. \tag{21}$$

Here $N_{\max}$ is the maximum image dimension and $\alpha = 1/2$. Each sample is taken from a smaller window until the sampling window becomes smaller that 3×3 pixels. This ensures that offsets close to the current candidate are more densely sampled. Taking $N_{\max}$ as the largest image dimension guarantees that all the image can be sampled with non-zero probability, which guarantees the asymptotic convergence of the algorithm to the true NNF.

The computational cost of one PatchMatch iteration (random search and propagation) is $\mathcal{O}(|\Omega_p||\widetilde{O}|)$, whereas the memory requirements are of $\mathcal{O}(|\widetilde{O}|)$. For most applications, a few iterations are often sufficient. See [6] and [7] for more details about the PatchMatch algorithm and some generalizations, and see [1] for an analysis of its convergence rate.

3.3.1 Parallelization of PatchMatch

Parallelization might be used to further accelerate the nearest neighbor field estimation with PatchMatch. In [6] it was proposed to implement PatchMatch on a GPU using the jump flood algorithm of [39] to accelerate the propagation step. Here, we focus on the parallelization for CPU considering several options.

Algorithm 4: PatchMatch parallelization with decoupling

```
...
for iter = 0, ..., I − 1 do
    // Offset propagation (raster or inverse raster order)
    for n = 0, ..., |Õ| − 1 do
        Propagate at pixel n ...
    // Random search. The following loop is parallel
    parallel for n = 0, ..., |Õ| − 1 do
        Random search at pixel n ...
```

Note in Algorithm 3 that at a given pixel the propagation step is immediately followed by the random search. The loops cannot be parallelized naively because the propagation is sequential: it depends on the preceding neighbors according to the propagation order. One approach would be to decouple the propagation and search steps as in Algorithm 4: the propagation is done first for all pixels and then the random search is performed on all pixels. In this way, all data dependencies are left in the propagation step and the search step can be easily parallelized. However, such decoupling slows down the propagation of good random offsets, therefore it affects the convergence speed (that is, to attain a similar result, Algorithm 4 may require more iterations than Algorithm 3).

A slightly different approach to parallelization considers the pixels diagonal-by-diagonal (going from the top-right to the bottom-left corner of the image) instead of the usual row-by-row. In this way, during the propagation step (on odd iterations), each pixel depends only on its top and left neighbors (or bottom and right neighbors on even iterations). This means, that pixels in the same diagonal are independent during both propagation and search steps, hence can be processed in parallel. This algorithm yields exactly the same result as Algorithm 3. However, since it violates the data locality principle, in practice it has a poor performance because of the reduced efficiency of data caching in the CPUs.

In this work we implement a parallelization scheme, mentioned in [7]. Assuming that P CPUs are available, we split the inpainting domain in chunks of equal sizes. In each iteration propagation and random search are performed independently for each chunk on a separate processor, as detailed in Algorithm 5. Each chunk consists of $|\widetilde{O}|/P$ pixels of the inpainting domain. In a row-by-row representation of an image each processor gets some adjacent rows. In this way, within a single iteration, a good offset cannot be propagated across chunks. To allow propagation across the boundary of a chunk we use two separate buffers for odd and even chunks and after each iteration we copy the boundary rows from one buffer to another. As a drawback, this scheme affects the convergence speed due to the limited propagation, however, to our knowledge this effect is negligible.

3.4 Multiscale Scheme

As in many state of the art exemplar-based inpainting methods (e.g. [26, 27, 43]), we incorporate a multiscale scheme. As noted in [24] and, as the example of Figure 4 suggests, inpainting is inherently a multiscale problem: images have structures of different sizes, ranging from large objects (as the lamps and their periodic distribution on the image) to fine scale textures (like the characters) and edges. The multiscale scheme responds to the fact that several patch sizes are needed to reproduce all these structures properly.

In Figure 4, we show completions obtained with patch NL-means using different patch sizes. We used a Gaussian intra-patch weight kernel g with standard deviation a. The figure shows two

Algorithm 5: PatchMatch parallelization by chunks

```
...
θ^0 ← Randomly initialize the entire offset map          // Create two output buffers
θ^1 ← Randomly initialize the entire offset map
for iter = 0, ..., I − 1 do
    // The following loop is parallel. nCPU is the number of CPUs or threads.
    parallel for r = 0, ..., nCPU − 1 do
        range ← [r |Õ|/nCPU, ..., (r + 1) |Õ|/nCPU − 1]      // Determine the range for the r-th CPU
                                                  // Process the range updating the correct output buffer
        for x ∈ range do
            θ_x^(r mod 2) ← ... Propagate ...
            θ_x^(r mod 2) ← ... Random search ...
    (θ^0_[odd ranges], θ^1_[even ranges]) ← (θ^1_[odd ranges], θ^0_[even ranges])   // Synchronize output buffers
θ ← θ^0                                                                            // Pick one buffer
```

results with a small patch ($a = 4$) and one result with a large patch ($a = 19$). The latter is able to reproduce the periodic pattern of the lamps, but the completion is blurry due to the spacial overlap of the patches and presents many discontinuities at the boundary of the inpainting domain.

The results with the small patch (second and fourth columns) do not show these artifacts, but one of them has failed to reproduce the lamps. The only difference between both is the initialization. The one in the second column was initialized with the original image, whereas the other one with the result obtained with the multiscale approach described in this section.

The implemented multiscale method goes along the lines of what is customary in the literature [22, 26, 43]. It consists in applying sequentially the inpainting scheme on a Gaussian image pyramid, starting at the coarsest scale. The result at each scale is upsampled and used as initialization for the next finer scale. The patch size is constant through scales.

Construction of a masked Gaussian pyramid. We create two pyramids: one for the image and one for the inpainting domain mask. As will be detailed soon, at any scale, for the image filtering we do not use the data inside the inpainting domain. Let us consider S scales, the finest denoted with $s = 0$. We will specify the size of the image at the coarsest level A_{S-1}. Denoting the size of the image at the finest scale by A_0, we compute the sampling rate as $r := (A_0/A_{S-1})^{1/(S-1)} \geqslant 1$. The width of the Gaussian filter σ is associated with the sub-sampling factor r as $\sigma(r) = 0.62\sqrt{r^2 - 1}$ (see [32] for details). Let a_0 be the size of the patch and $\mathcal{E}_{a_0}$ the corresponding energy. We will add the superindex $s = 0, \ldots, S-1$ to the variables u and φ to denote the scale. As before, the subindex 0 refers to the initial condition, i.e. u_0^s is the initial condition at scale s.

The mask pyramid is created by applying a fixed threshold T_O after each downsampling step. Thus, we have the following relation between the characteristic functions of the inpainting domains O^s and O^{s+1}:

$$\chi_{O^{s+1}} = (\downarrow_r g_{\sigma(r)} * \chi_{O^s}) > T_O.$$

The value for T_O is set to 0.4, which implies that the inpainting domain is shrunk a little from one scale to the next. Given a scalar image f and a scalar T, by $f > T$ we denote a binary image which is one if and only if $f(x) > T$.

To generate the image pyramid, we use masked convolutions that only use the data in the known

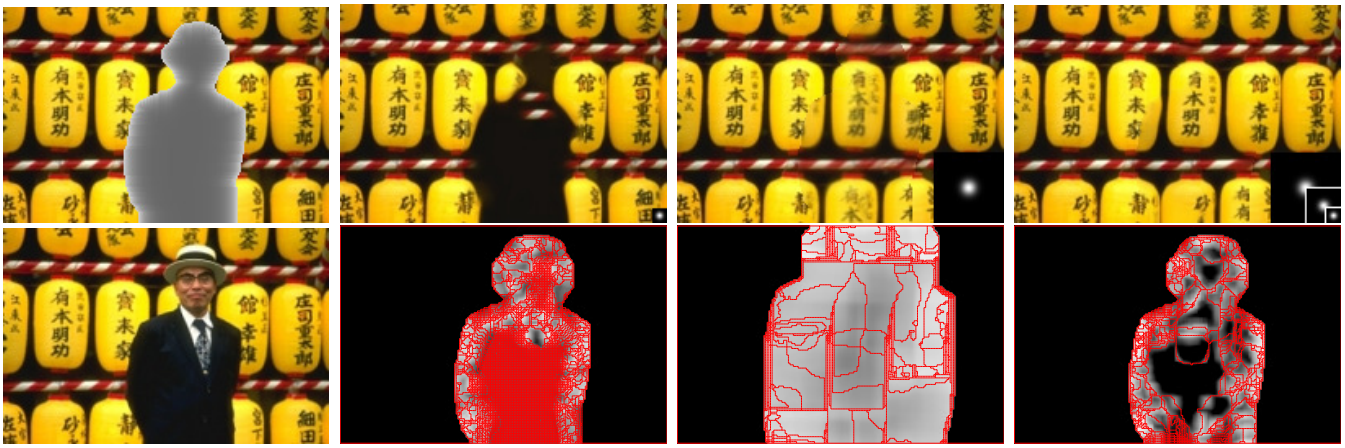


Figure 4: **Single scale vs. multiscale.** Left column: inpainting domain and initial condition. For the rest of the columns, from left to right: single scale inpainting with a 9×9 patch with $a = 4$; single scale inpainting with a 43×43 patch with $a = 19$; multiscale inpainting with three scales, corresponding to patch sizes of 9×9 with $a = 4$, 21×21 with $a = 9$, and 43×43 with $a = 19$. All results have been computed with the patch NL-means scheme. The bottom row shows the boundaries between copy regions superimposed over the energy density image.

part of the image. At scale s, we have that

$$u_0^{s+1} = \downarrow_r \text{masked-convolution}(g_{\sigma(r)}, (O^s)^c; u^s),$$

The masked convolution operator, is defined for images g and f and a region A, as

$$\text{masked-convolution}(g, A; f)(x) = \frac{\sum_{h\in\mathbb{Z}^2} g(x-h)\chi_A(h)f(h)}{\sum_{h\in\mathbb{Z}^2} g(x-h)\chi_A(h)} = \frac{g * (\chi_A f)(x)}{g * \chi_A(x)}.$$

In other words, the image at scale s is multiplied by the mask of known data before the convolution with the smoothing kernel. Afterwards a normalization is applied to each pixel.

Joint image and NNF upscaling. When going from scale $s+1$ to s we need to upsample both the correspondence map and the inpainted image. We proceed as in [43]. The coarse correspondences φ^{s+1} are first interpolated to the finer image size, yielding $\uparrow_r \varphi^{s+1}$. For this we use nearest neighbor interpolation. These interpolated correspondences are then scaled by r, yielding $\varphi_0^s = r \cdot \uparrow_r \varphi^{s+1}$. The image is upsampled by solving an image update step at the finer scale, using the upsampled correspondences: $u_0^s = \min_u \mathcal{E}_{a_0}(u, \varphi_0^s)$. More conventional upsampling schemes by local interpolation (such as bilinear or splines) introduce a bias towards low-frequency non-textured regions. This exemplar-based upsampling avoids such bias.

Notice that keeping the patch size constant while filtering and reducing the image can be seen as the sequential minimization of a series of inpainting energies with varying patch size given by $a_s = (1/r^s)a_0$, $s = 0, \ldots, S-1$, over a corresponding series of filtered images. In the coarsest scale $S-1$, a larger portion of the inpainting domain is covered by partially known patches. This makes the inpainting task easier and less dependent on the initialization. The energy at this scale should have fewer local minima. The dependency of the minimization process on the initial condition ensures that each single scale solution remains close to the coarse scale initialization. The multiscale algorithm *exploits* this dependency to obtain an image u^0 which is approximately self-similar for all scales (or equivalently, for all patch sizes).

Figure 4 shows a comparison between single and multiscale results with the patch NL-means scheme. The multiscale result shows the benefits of large and small patch sizes. The missing lamps

Algorithm 6: Multiscale inpainting scheme

input : known image u_0, inpainting domain O, patch size a_0, number of scales S, size of coarsest scale A_{S-1}
output: inpainted image u^s and offset map φ^s at each scale

$r \leftarrow (\text{size}(u_0)/A_{S-1})^{1/(S-1)}$ *// size factor*

$\{O^s\}_{s=0,\dots,S-1} \leftarrow$ Compute pyramid of domains (S scales, factor r) from O
$\{u_0^s\}_{s=0,\dots,S-1} \leftarrow$ Initialize pyramid of images (S scales, factor r) from u_0 and O

for $s = S-2, \dots, 0$ **do**
  if $s = S-1$ **then**
    $u_0^s \leftarrow$ Initialize inpainting domain with average value of u_0 in $\widetilde{O}^c$
  else
    $\varphi_0^s \leftarrow r \cdot (\uparrow_r \varphi^{s+1})$ *// upsample using nearest neighbor interpolation and scale by r*
    $u_0^s \leftarrow \arg\min_u \mathcal{E}_{a_0}(u, \varphi_0^s)$ *// solve image update (§3.2)*
      s.t. $u(x) = u_0^s(x)$, $\forall x \notin O^s$

  $(u^s, \varphi^s) \leftarrow \arg\min_{(u,\varphi)} \mathcal{E}_{a_0}(u, \varphi)$ *// call to Algorithm 1*
    s.t. $u(x) = u_0^s(x)$, $\forall x \notin O^s$,
    with initial condition u_0^s in O^s

have been completed with the correct shape and spacing by the coarser stages, and the fine details are overall much less blurry. Also, there are almost no discontinuities at the boundary of O. The bottom row shows the copy regions. The single scale results show a coarse partition for the large patch (the copying is more rigid), and one with many small regions for the smaller patch. The multiscale NNF shows an intermediate partition, with some large regions inside of the hole and smaller ones around its boundary. The inpainting at the finer scales works by refining the coarse partition obtained at coarser scales.

3.5 Confidence Mask

For large inpainting domains, it is useful to introduce a mask $\kappa : \Omega \to (0, 1]$ which assigns a confidence value to each pixel, depending on the certainty of its information (see also [16, 27]). This helps in guiding the flow of information from the boundary towards the interior of the hole, eliminating some local minima and reducing the effect of the initial condition. The resulting energy takes the form

$$\mathcal{E}_E(u, \varphi) = \sum_{x \in \widetilde{O}} \kappa(x) E(p_u(x) - p_{\hat{u}}(\varphi(x))). \tag{22}$$

The effect of κ on the image update step can be seen on the pixel influence weights m

$$m_{z\hat{z}} = \sum_{h \in \Omega_p} g_h \kappa(z-h) \delta_{\varphi(z-h)}(\hat{z}-h) \chi_{\widetilde{O}}(\hat{z}-h).$$

Thus, the contribution of the patch $p_u(z-y)$ to the evidence function is now weighted by its confidence. Patches with higher confidence will have a stronger influence.

For the experiments shown in this paper, the confidence mask was set to

$$\kappa(x) = \begin{cases} (1 - c_0) \exp\left(-\frac{d(x,\partial O)}{t_c}\right) + c_0 & \text{if } x \in O, \\ 1 & \text{if } x \in O^c, \end{cases}$$

which shows an exponential decay w.r.t. the distance to the boundary inside the inpainting domain $d(\cdot, \partial O)$. Here $t_c > 0$ is the decay time and $c_0 > 0$ determines the asymptotic value reached far away from the boundary. Setting $t_c = 0$ results in a constant confidence mask.

3.6 Color Images

Energies for color images are obtained by defining a patch error function for color patches as the sum of the error functions of the three scalar components

$$E(p_{\mathbf{u}}(x) - p_{\hat{\mathbf{u}}}(\hat{x})) = \sum_{i=1}^{3} E(p_{u_i}(x) - p_{\hat{u}_i}(\hat{x})),$$

where $\mathbf{u} : \Omega \to \mathbb{R}^3$ is the color image, and u_i (with $i = 1, 2, 3$) are its components (analogously for gradient-based errors). Given the correspondences, each channel is updated using the scheme for scalar images. All channels are updated using the same correspondences.

4 Experiments and Parameter Selection

In this section we demonstrate the proposed schemes on real inpainting problems and compare them with four representative state of the art methods. The images used were obtained from [27] and from the inpainting benchmark proposed by [26], available at `http://yokoya.naist.jp/research/inpainting/`.

4.1 Experimental Setting

We consider three inpainting methods, variations of our proposed framework, namely patch NL-means, -medians and -Poisson. In all cases we use the multiscale approach and the CIELab color space.

Some representative results are shown in Figure 5. Obtaining good results requires fixing the following parameters (see Table 1 for the concrete values used in the experiments):

Patch size. We used square patches with constant intra-patch weights ($g = 1/|\Omega_p|$). Gaussian-weighted patches require a larger support, reducing the available exemplars.

Multiscale parameters. The multiscale scheme has two parameters: the size of the coarsest image A_{S-1} and the number of scales S. The former is the most critical one. We specify A_{S-1} as a percentage of the original size. The default value is set according to [34, 33], where it is noted that the size of the inpainting domain in the coarsest scale should be smaller than twice the patch size. In our implementation, the default value for A_{S-1} is

$$A_{S-1} = \frac{1.5 \cdot \text{patch size}}{\max_{x \in O} d(x, O^c)}.$$

This default value can be overridden by the user. A reasonable initial guess for the number of scales S can be such that the subsampling rate $r = (A_0/A_{S-1})^{1/(S-1)} \approx 2^{1/3} \approx 1/0.8$ as in [43].

	Matsuri			Mailboxes			Baseball			Sofa		
	M	Md	P	M	Md	P	M	Md	P	M	Md	P
S	10	10	11	7	7	7	6	6	6	8	6	8
A_{S-1},(%)	15	10	10	30	50	30	30	30	30	20	50	20
patch size	7	7	7	9	11	9	9	9	7	11	11	7
t_c	5	3	5	5	3	4	5	5	5	5	5	3
λ	–	–	0.01	–	–	0.05	–	–	0.01	–	–	0.01

Table 1: Parameters for the results on the images in Figures 4 and 5.

Confidence mask. The confidence mask has two parameters, the asymptotic value c_0 and the decay time t_c. For all experiments we fix $c_0 = 0.1$.

Initialization. To initialize the inpainting domain in the coarsest scale, we used a (local) Poisson interpolation by solving $\Delta u(x) = 0$ with Dirichlet boundary conditions.

Mixing coefficient. For the patch NL-Poisson scheme the mixing coefficient λ should be specified as well. Recall that lower values of λ give a higher weight to the gradient component of the energy. This is appropriate for structured images with strong edges.

For the sake of comparison we also include results obtained using four representative methods from the state of the art. Two of them compute a coarse correspondence map, the *PatchWorks* (PW) method [35] and the approach of [27] (KT). The first one is greedy and the latter is iterative. Seams between blocks are eliminated a posteriori, using for instance a Poisson blending. The other two methods compute a dense correspondence map: *Resynthetizer* (R) [23] (greedy) and the approach of [26] (KSY) (iterative). The latter is similar to the patch NL-means, with two improvements: locality of the nearest neighbor search and a correction that accounts for multiplicative brightness changes.

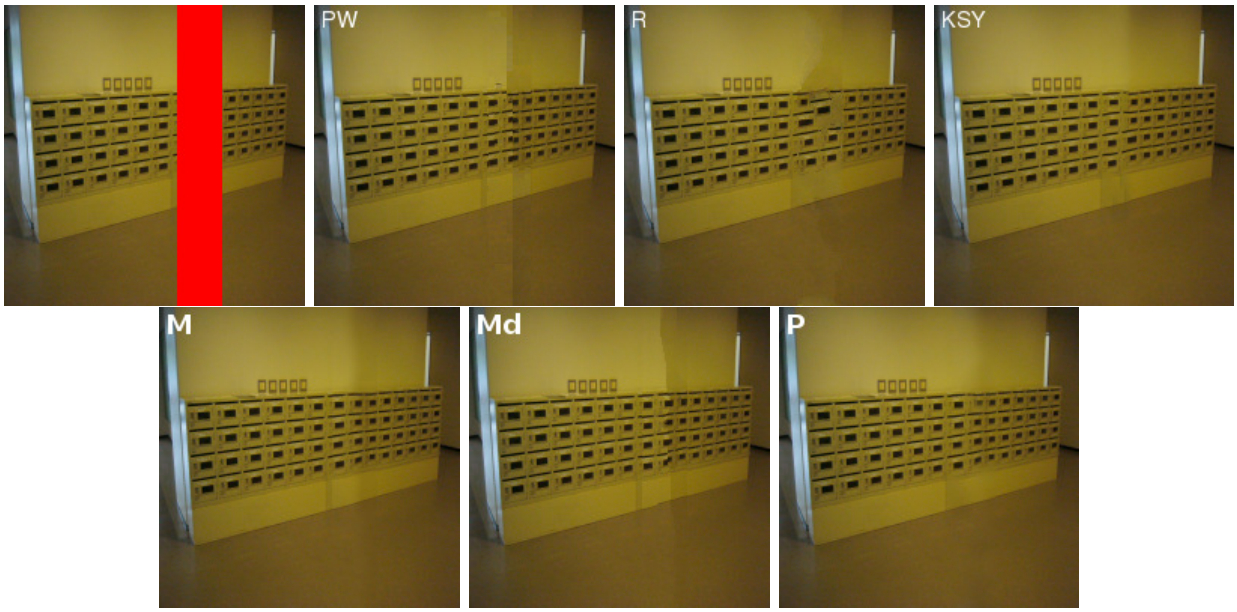
The results from KT were published in [27] and those from KSY can be found in [26] and are also available at `http://yokoya.naist.jp/research/inpainting/`. The Resynthetizer algorithm is implemented as a plug-in for the GIMP image editing software[2]. Our implementation of PatchWorks was written by Geoffrey Scoutheeten and was kindly made available to us by Simon Masnou. It does not include the blending post-processing step, so all seams are visible. We refer the reader to [35] and [13] for results obtained using this efficient technique with blending.

The three inpainting schemes presented here differ in the way the patches are blended. Methods based on the L^2-norm perform a smooth blending, whereas those based on the L^1-norm favor sharper transitions.

It is notorious that patch NL-medians performs better at reproducing fine textures (Figure 5b). The results of the L^2 methods are smoothed by the spatial averaging of overlapping patches. On the other hand, patch NL-medians creates sharp discontinuities when different copy regions meet (Figure 5c). These discontinuities are very noticeable and in these cases some smoothing is desirable. Also patch NL-medians generally requires the use of larger patches, particularly for structured images.

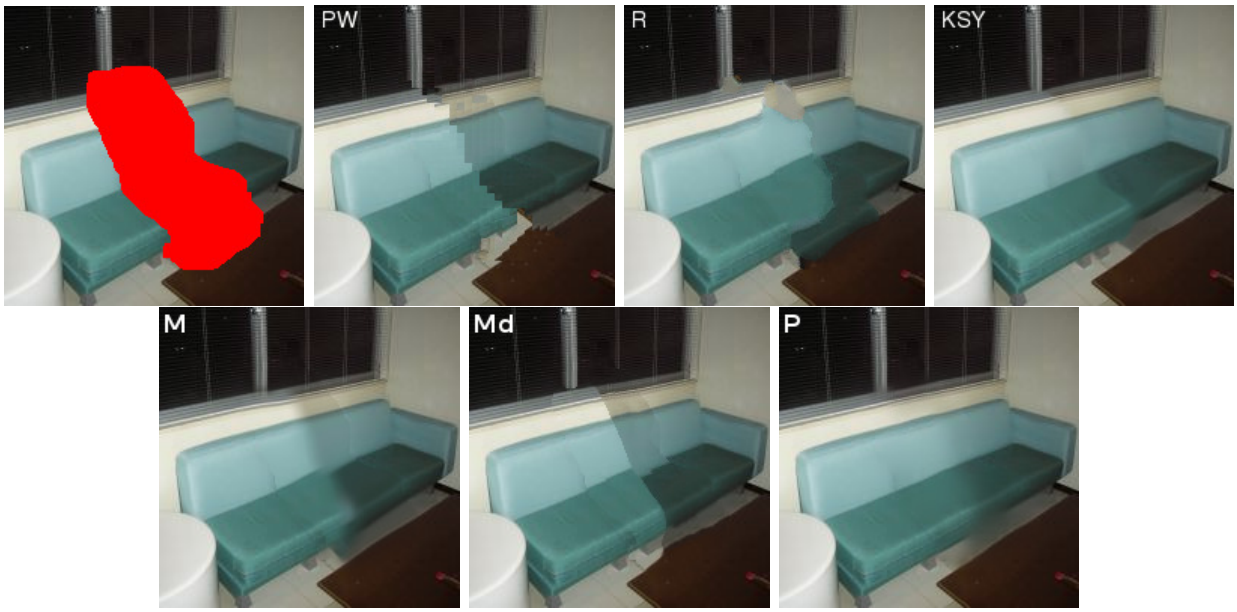
Patch NL-Poisson combines the exemplar-based interpolation with PDE-based diffusion schemes. This results in a smoother continuation of the information across the boundary and inside the inpainting domain, and in a better propagation of structures (Figures 5c and 5a). Furthermore, the inclusion of gradients in the patch similarity criterion allows to handle additive brightness changes.

[2]GIMP: GNU Image Manipulation Program. S. Kimball, P. Mattis and the GIMP Dev. Team. `http://www.gimp.org/`. Version 2.6.8 released on December 2009.



(a) *Example with periodic textures (Mailboxes)*

(b) *Example with random textures (Baseball)*



(c) *Example with structures (Sofa)*

Figure 5: **Results on periodic, textured, and structured images.** PW: PatchWorks [35], R: Resynthesizer [23] ,KT: method of [27], and KSY: Kawai et al. [26]. M, Md and P stand for patch NL-means, -medians and -Poisson.

This is not the case for intensity-based methods (patch NL-means and NL-medians), which produce abrupt transitions at the boundary of the hole and between copy regions (Figures 5a and 5c).

A Organization of the Code

In this section we briefly describe the architecture of the code. The provided implementation is written in C++ and is object oriented, so the algorithm is encapsulated in a number of classes, which can be divided into data structures, algorithms, and utilities. The code was designed to be modular so that these new schemes can be added easily: only the patch similarity criterion and the corresponding image update step need to be implemented.

Although the code is written with more emphasis on readability than on performance, for the computation of the correspondence map we implemented a parallel version of PatchMatch [6].

A.1 Data Structures

These classes were designed to store the data:

Shape: container for a 2D rectangular shape (width and height).

Image: container for an image with some defined pixel type.

Mask: container for a binary mask. Mask differs from image in that it provides methods to iterate over the region represented by the mask.

A.2 Algorithms

These are the main classes that encapsulate the logic of the inpainting algorithms:

ImageInpainting: the main class implementing the image inpainting method. It should be parameterized by the PatchMatch class and a class derived from AImageUpdating.

PatchMatch: implementation of the Patch-Match algorithm. It should be parameterized by a class derived from APatchDistance.

APatchDistance: abstract base class for different patch distances. We have implemented the following patch distances, derived from APatchDistance:

- **L2NormPatchDistance:** implementation of the L2-Norm patch distance.
- **L1NormPatchDistance:** implementation of the L1-Norm patch distance.
- **L2CombinedPatchDistance:** implementation of the L2-Norm patch distance which takes into account intensity differences and gradient differences. The relative importance of the intensity term over the gradient term is controlled by the 'lambda' parameter.

AImageUpdating: abstract base class for different image updating methods. We have implemented the following image update methods, derived from AImageUpdating:

- **PatchNonLocalPoisson:** implementation of the Non-Local Poisson image updating scheme.
- **PatchNonLocalMeans:** implementation of the Non-Local Means image updating scheme.
- **PatchNonLocalMedians:** implementation of the Non-Local Medians image updating scheme.

Abstract APatchDistance and AImageUpdating classes with their corresponding derived classes allow to easily configure the algorithm to apply any desired approach (NL-means, NL-Poisson, etc.). The following example shows how they are used.

A.2.1 Usage Example

Setup Image inpainting algorithm.

```
// setup patchmatch
Shape patch_size = Shape(patch_side, patch_side);
APatchDistance *patch_distance    = new L2CombinedPatchDistance(lambda,
                                    patch_size, patch_sigma);
PatchMatch *patch_match           = new PatchMatch(patch_distance,
                                    patch_match_iterations, random_shots_limit, -1);

// setup image update
AImageUpdating *image_updating    = new PatchNonLocalPoisson(patch_size, patch_sigma,
                                    lambda, conj_grad_tol, conj_grad_iter);

// create inpainting object
ImageInpainting image_inpainting = ImageInpainting(inpainting_iterations,
                                    tolerance,
                                    scales_amount,
                                    subsampling_rate,
                                    confidence_decay_time,
                                    confidence_asymptotic_value,
                                    initialization_type);      // enum

image_inpainting.set_weights_updating(patch_match);
image_inpainting.set_image_updating(image_updating);
```

Call the algorithm.

```
// image of type Image, and mask of type Mask
Image<float> output = image_inpainting.process(input, mask);
```

The implementation can be extended to new inpainting schemes corresponding to different patch similarity criteria. To do so, one needs to implement only the image update class and the patch distance class.

Acknowledgment

Work partly founded by the Centre National d'Etudes Spatiales (CNES, MISS Project), BPIFrance and Rgion Ile de France, in the framework of the FUI 18 Plein Phare project, the European Research Council (advanced grant Twelve Labours 246961), and the Office of Naval research (ONR grant N00014-14-1-0023).

Image Credits

 The Berkeley Segmentation Dataset and Benchmark[3]

 The Berkeley Segmentation Dataset and Benchmark [3]

 The Berkeley Segmentation Dataset and Benchmark [4]

 Norihiko Kawai [26], Evaluation for Image Inpainting image No. 057[5]

[3] http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/
[5] http://yokoya.naist.jp/research/inpainting/

 Norihiko Kawai [26], Evaluation for Image Inpainting image No. 078[5]

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. http://dx.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. http://dx.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 of Mathematical Analysis, 42 (2010), pp. 1246–1285. http://dx.doi.org/10.1137/080743883.

[4] S. P. Awate and R. T. Whitaker, *Unsupervised, information-theoretic, adaptive image filtering for image restoration*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 28 (2006), pp. 364–376. http://dx.doi.org/10.1109/TPAMI.2006.64.

[5] C. Ballester, M. Bertalmío, V. Caselles, G. Sapiro, and J. Verdera, *Filling-in by joint interpolation of vector fields and gray levels*, IEEE Transactions on Image Processing, 10 (2001), pp. 1200–1211. http://dx.doi.org/10.1109/83.935036.

[6] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, *PatchMatch: a randomized correspondence algorithm for structural image editing*, in Proceedings of SIGGRAPH, New York, NY, USA, 2009, ACM, pp. 1–11. http://dx.doi.org/10.1145/1531326.1531330.

[7] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein, *The generalized PatchMatch correspondence algorithm*, in European Conference on Computer Vision, 2010. http://dx.doi.org/10.1007/978-3-642-15558-1_3.

[8] M. Bertalmío, G. Sapiro, V. Caselles, and C. Ballester, *Image inpainting*, in Proceedings of SIGGRAPH, New York, NY, USA, 2000, ACM. http://dx.doi.org/10.1145/344779.344972.

[9] M. Bertalmío, L. Vese, G. Sapiro, and S. J. Osher, *Simultaneous structure and texture inpainting*, IEEE Transactions on Image Processing, 12 (2003), pp. 882–889. http://dx.doi.org/10.1109/TIP.2003.815261.

[10] R. Bornard, E. Lecan, L. Laborelli, and J.-H. Chenot, *Missing data correction in still images and image sequences*, in Proceedings ACM International Conference on Multimedia, 2002. http://dx.doi.org/10.1145/641007.641084.

[11] F. Bornemann and T. März, *Fast image inpainting based on coherence transport*, Journal of Mathematical Imaging and Vision, 28 (2007), pp. 259–278. http://dx.doi.org/10.1007/s10851-007-0017-6.

[12] A. Buades, B. Coll, and J.-M. Morel, *A non local algorithm for image denoising*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, 2005, pp. 60–65. http://dx.doi.org/10.1109/CVPR.2005.38.

[13] F. Cao, Y. Gousseau, S. Masnou, and P. Pérez, *Geometrically guided exemplar-based inpainting*, SIAM Journal on Imaging Sciences, 4 (2011), pp. 1143–1179. http://dx.doi.org/10.1137/110823572.

[14] T. Chan, S. H. Kang, and J. H. Shen, *Euler's elastica and curvature based inpaintings*, SIAM Journal of Applied Mathematics, 63 (2002), pp. 564–592. http://dx.doi.org/10.1137/S0036139901390088.

[15] T. Chan and J. H. Shen, *Mathematical models for local nontexture inpaintings*, SIAM Journal of Applied Mathematics, 62 (2001), pp. 1019–1043. http://dx.doi.org/10.1137/S0036139900368844.

[16] A. Criminisi, P. Pérez, and K. Toyama, *Region filling and object removal by exemplar-based inpainting*, IEEE Transactions on Image Processing, 13 (2004), pp. 1200–1212. http://dx.doi.org/10.1109/TIP.2004.833105.

[17] L. Demanet, B. Song, and T. Chan, *Image inpainting by correspondence maps: a deterministic approach*, Applied and Computational Mathematics, 1100 (2003), pp. 217–250. http://math.stanford.edu/~laurent/papers/demanet_song_chan.pdf.

[18] I. Drori, D. Cohen-Or, and H. Yeshurun, *Fragment-based image completion*, ACM Transactions on Graphics. Special issue: Proceedings of ACM SIGGRAPH, 22 (2003), pp. 303–312. http://dx.doi.org/10.1145/1201775.882267.

[19] A. A. Efros and T. K. Leung, *Texture synthesis by non-parametric sampling*, in Proceedings of the IEEE Internatonal Conference in Computer Vision, September 1999, pp. 1033–1038. http://dx.doi.org/10.1109/ICCV.1999.790383.

[20] S. Esedoglu and J. H. Shen, *Digital image inpainting by the Mumford-Shah-Euler image model*, European Journal of Applied Mathematics, 13 (2002), pp. 353–370. http://dx.doi.org/10.1017/S0956792502004904.

[21] G. Facciolo, P. Arias, V. Caselles, and G. Sapiro, *Exemplar-based interpolation of sparsely sampled images*, in EMMCVPR, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 331–344. http://dx.doi.org/10.1007/978-3-642-03641-5_25.

[22] C.-W. Fang and J.-J. J. Lien, *Rapid image completion system using multiresolution patch-based directional and nondirectional approaches*, IEEE Transactions on Image Processing, 18 (2009), pp. 2769–2779. http://dx.doi.org/10.1109/TIP.2009.2027635.

[23] P. Harrison, *Texture tools.*, PhD thesis, Monash University, 2005. http://www.logarithmic.net/pfh-files/thesis/dissertation.pdf.

[24] M. Holtzman-Gazit and I. Yavneh, *A scale-consistent approach to image completion*, International Journal on Multiscale Computational Engineering, 6 (2008), pp. 617–628. http://dx.doi.org/10.1615/IntJMultCompEng.v6.i6.80.

[25] H. Igehy and L. Pereira, *Image replacement through texture synthesis*, in Proceedings of the IEEE International Conference on Image Processing, October 1997. http://dx.doi.org/10.1109/ICIP.1997.632049.

[26] N. Kawai, T. Sato, and N. Yokoya, *Image inpainting considering brightness change and spatial locality of textures and its evaluation*, in Advances in Image and Video Technology, Springer Berlin Heidelberg, 2009, pp. 271–282. http://dx.doi.org/10.1007/978-3-540-92957-4_24.

[27] N. Komodakis and G. Tziritas, *Image completion using efficient belief propagation via priority scheduling and dynamic pruning*, IEEE Transactions on Image Processing, 16 (2007), pp. 2649–2661. http://dx.doi.org/10.1109/TIP.2007.906269.

[28] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, *Texture optimization for example-based synthesis*, ACM Transactions on Graphics, 24 (2005), pp. 795–802. http://dx.doi.org/10.1145/1186822.1073263.

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

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

[31] S. Masnou and J.-M. Morel, *Level lines based disocclusion*, in Proceedings of IEEE International Conference on Image Processing, 1998. http://dx.doi.org/10.1109/ICIP.1998.999016.

[32] J.-M. Morel and G. Yu, *On the consistency of the SIFT Method*, Preprint, CMLA, 26 (2008). http://www.ipol.im/pub/art/2011/my-asift/SIFTconsistency.pdf.

[33] A. Newson, *On video completion: line scratch detection in films and video inpainting of complex scenes*, PhD thesis, Telecom ParisTech, 2014. https://tel.archives-ouvertes.fr/tel-01136253/.

[34] 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. http://dx.doi.org/10.1137/140954933.

[35] P. Pérez, M. Gangnet, and A. Blake, *PatchWorks: Example-based region tiling for image editing*, tech. report, Microsoft Research, 2004. http://research.microsoft.com/apps/pubs/default.aspx?id=70036.

[36] G. Peyré, *Manifold models for signals and images*, Computer Vision and Image Understanding, 113 (2009), pp. 249–260. http://dx.doi.org/10.1016/j.cviu.2008.09.003.

[37] G. Peyré, S. Bougleux, and L. D. Cohen, *Non-local regularization of inverse problems*, Inverse Problems and Imaging, 5 (2011), pp. 511–530. http://dx.doi.org/10.3934/ipi.2011.5.511.

[38] Y. Pritch, E. Kav-Venaki, and S. Peleg, *Shift-map image editing*, in Proceedings of the 12th International Conference in Computer Vision, Kyoto, Sept 2009, pp. 151–158. http://dx.doi.org/10.1109/ICCV.2009.5459159.

[39] G. Rong and T.-S. Tan, *Jump flooding in GPU with applications to Voronoi diagram and distance transform*, in Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games, I3D '06, New York, NY, USA, 2006, ACM, pp. 109–116. http://doi.acm.org/10.1145/1111411.1111431.

[40] J.R. Shewchuk, *An introduction to the conjugate gradient method without the agonizing pain*, tech. report, Pittsburgh, PA, USA, 1994. http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf.

[41] D. Tschumperlé and R. Deriche, *Vector-valued image regularization with PDE's: a common framework for different applications*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 27 (2005). http://dx.doi.org/10.1109/CVPR.2003.1211415.

[42] L.-Y. Wei and M. Levoy, *Fast texture synthesis using tree-structured vector quantization*, in Proceedings of the SIGGRAPH, New York, NY, USA, 2000, ACM. http://dx.doi.org/10.1145/344779.345009.

[43] 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. http://dx.doi.org/10.1109/TPAMI.2007.60.