Exemplar-Based Inpainting: Technical Review and New Heuristics for Better Geometric Reconstructions

Article *in* IEEE Transactions on Image Processing · March 2015

DOI: 10.1109/TIP.2015.2411437 · Source: PubMed

4 authors:

Pierre Buyssens
Université de Caen Normandie
36 PUBLICATIONS **320** CITATIONS
SEE PROFILE

Maxime Daisy
Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de C…
16 PUBLICATIONS **200** CITATIONS
SEE PROFILE

David Tschumperlé
French National Centre for Scientific Research
85 PUBLICATIONS **3,306** CITATIONS
SEE PROFILE

Olivier Lezoray
Université de Caen Normandie
256 PUBLICATIONS **2,986** CITATIONS
SEE PROFILE

Exemplar-based Inpainting: Technical Review and new Heuristics for better Geometric Reconstructions

P. Buyssens*, M. Daisy*, D. Tschumperlé, O. Lézoray

***Abstract*—This paper proposes a technical review of *exemplar-based* inpainting approaches with a particular focus on greedy methods. Several comparative and illustrative experiments are provided to deeply explore and enlighten these methods, and to have a better understanding on the state-of-the-art improvements of these approaches. From this analysis three improvements over Criminisi *et al.* algorithm are then presented: 1) a tensor-based data term for a better selection of pixel candidates to fill in, 2) a fast patch lookup strategy to ensure a better global coherence of the reconstruction, and 3) a fast anisotropic spatial blending algorithm that reduces typical block artifacts using tensor models. Relevant comparisons to state-of-the-art inpainting methods are provided that exhibit the effectiveness of our contributions.**

***Index Terms*—Exemplar-based image inpainting, Structure tensor analysis, Patch lookup strategy, Anisotropic spatial blending**

I. Introduction and context

Image inpainting, also known as "disocclusion", "filling-in" or "image completion", refers to the recovery of missing or damaged parts in an image such that the resulting image looks as natural as possible.

It differs from other common image processing techniques such as denoising since the part to inpaint (often referred to *mask* or *hole*) is completely unknown. Numerous applications for inpainting have emerged over recent years: restoring images with scratches, text or logo removal, interpolation, and more complex ones such as removal of objects or persons.

Since 2000, intensive research on this topic has led to various and numerous inpainting algorithms. A general over-review of these has recently been proposed in [1]. They can be roughly divided into two main kinds of approaches: *geometry-based*, and *patch-based* methods. Although *geometry-based* approaches are beyond the scope of this paper, we briefly summarize them in Section II since they are the earlier inpainting methods proposed in the literature, and they inspired some *hybrid* methods described in the Section III. *Patch-based* methods use patches for the *analysis* of the image content, and include *sparsity-based* methods and the subset of *exemplar-based* approaches. Contrary to *sparsity-based* methods, the *exemplar-based* ones also use image content (i.e., patches) to reconstruct the missing data. They mainly proceed in a copy/paste fashion of patches chunks, or by mixing several patches together in a pixel-level inpainting scheme.

Our contributions are twofold:

- First we propose a review of *exemplar-based* inpainting methods. The seminal algorithm of Crimini *et al.* [2] is detailed and a technical review of up-to-date derived works is proposed.
- Second, we propose to develop and gather our previous adjustments [3], [4] of this algorithm into a unified framework. Particularly we propose in this paper a tensor-directed extension of our previous isotropic patch blending algorithm [3] that reduces typical recontruction artifacts of greedy inpainting methods.

Before entering the review of exemplar-based methods, we provide in Section II a taxonomy of image inpainting techniques by briefly summarizing *geometry-based* and *sparsity-based* methods. We also take a short detour into the subject of texture synthesis because important considerations of *exemplar-based* methods come from this domain.

Then we propose in Section III a technical review of *examplar-based* inpainting methods. We feature in Section IV three different adjustments to the original method we made, and show that they clearly improve the geometric coherence of the inpainted results. Each of these modifications is compared to appropriate state-of-the-art methods, and the visual improvements are shown in Section V on several difficult real cases examples. Section VI concludes the paper.

II. Inpainting methods taxonomy

A. Geometry-based methods

The notion of *inpainting* can be closely related to the notion of *interpolation* since both aim at determining unknown values of a function from known ones. Various types of *Geometry-aware* interpolation of images were the first introduced methods, and have been proposed in [5]–[13] using various different techniques: level lines completion in [6], anisotropic diffusion using variational or Partial Differential Equations (PDE's) in [5], [7], [12], minimization of the Total Variation [10] using curvature [8], coherence transport [13], and minimization of some image-based statistic functional [11], [14], [15]. These techniques introduce smoothness priors and tend to propagate local structures from the exterior of the mask to the interior.

They usually provide interesting results in terms of global geometry consistency. However, they mainly fail at reconstructing large-scale textures and tend to reconstruct flat-looking images. This kind of methods are then better adapted

∗ These authors contributed equally.

The authors are with the GREYC laboratory, Université de Caen Basse-Normandie, ENSICAEN, Caen, France.

Email: pierre.buyssens@unicaen.fr, maxime.daisy@ensicaen.fr, david.tschumperle@ensicaen.fr, olivier.lezoray@unicaen.fr

This research was supported by French national grant Action 3DS.

to small occlusions rather than large ones, or to process images that are continuous by parts (cartoon-like). They are sometimes used as a component of more complex hybrid inpainting schemes as detailed in further sections.

B. Sparsity-based methods

Sparsity-based inpainting methods take benefit of the natural redundancy present in an image under a given transformation [16]–[19]. They assume that the image can be sparsely represented in a given basis (wavelet, framelet, DCT ...), and the synthesis of the missing part is then optimized according to this sparse distribution. Depending on the actual content of the image (edges, textures, ...), the choice of the dictionnary can be critical. In [16], an image is first decomposed into *cartoon* and *textures* components. The inpainting of both is processed respectively with piecewise linear polynomial functions, and a DCT basis. As shown in Section III-C, such separation are also used in *hybrid methods*.

The sparsity-based inpainting approaches have been mainly derived from image denoising methods [20], and can then show impressive results [18] when the missing part is small, thin, or sparsely distributed over the image (like noise). Anyway, they mainly fail in reconstructing points of the image containing macro-textures.

C. Texture synthesis

Texture synthesis is important for many applications in computer graphics, vision, and image processing. Beyond the traditional stochastic model using Markov Random Fields [21], Efros and Leung [22] have been the first to introduce the notion of image patches for texture synthesis. The core of their proposed method is the concept of the self-similarity prior. This kind of prior has largely influenced recent and successfully advances in image processing [23].

The method proposed in [22] synthesizes a texture by estimating the value of a pixel p according to a patch Ψ_p centered on it. It is found by searching the k-nearest neighbors of Ψ_p according to the metric $d = d_{SSD} * G$ where d_{SSD} is the vector of squared differences and G is a $2D$ Gaussian weighting kernel. Wei *et al.* [24] have proposed a similar approach within a multiscale framework and focus on using a tree-structure vector quantization to speed up the patch search. M. Ashikhmin [25] further has proposed a smart search scheme that enhance the texture synthesis while being faster. This last method in fact influences our patch search scheme presented in Section IV-B2.

D. Exemplar-based methods

Many of the ideas introduced in texture synthesis have led to the so-called *exemplar-based* inpainting approaches more detailed in Section III, and in fact some inpainting-like results can already be found in [22]. The introduction of patches led to many new insights in the image processing field, and particularly in inpainting. In the early 2000s, Bornard *et al.* [26], Drori *et al.* [27], and Criminisi *et al.* [2] proposed independently three exemplar-based inpainting approaches that lay the foundation of important notions of the exemplar-based inpainting approaches. Many methods have then been derived from these seminal works, that we propose to classify and review hereafter.

III. A comprehensive review of exemplar-based inpainting methods

Exemplar-based inpainting techniques use patches of an image both to *analyze* its content to find the best way to inpaint it, and to *reconstruct* the missing part. Such methods can themselves be subdivided into three main approaches:

- the *greedy* approaches [2], [26]–[31] inpaint the hole in one pass by copying multiple patches or chunks in a greedy manner,
- the *hybrid* ones [32]–[35] that incorporate elements from the geometry-based methods to inpaint macro structures first,
- and the *energy-based* [36]–[42] methods that often minimizes an energy and requires several iteration to converge. These methods often reconstruct the missing part at the pixel level by mixing patches together.

A. Notations and definitions

An image to be inpainted is considered as a function $I : \mathcal{I} \rightarrow \mathbb{R}^3$ (color image) where $\mathcal{I}$ defines the image domain, Ω is the masked part of the image (i.e., the unknown pixels to resynthesize), and $\delta\Omega$ is the boundary of the mask. In the following, a patch Ψ_p centered on the pixel p is considered as a function $\Psi_p : \mathcal{N}_p \rightarrow \mathbb{R}^3$ where $\mathcal{N}_p \subset \mathcal{I}$ is the square support of Ψ_p. Note that this patch can be masked (i.e., some of its pixels are unknown). $\Psi_{\hat{p}}$ denotes a patch that matches Ψ_p according to a given metric d:

$$\Psi_{\hat{p}} \;=\; \left\{ \Psi_q \middle| \underset{q|\mathcal{N}_q \cap (\mathcal{I}-\Omega)}{\arg\min}\; d(\Psi_p, \Psi_q) \right\}$$

The simplest (and widely used essentially for computational efficiency purposes) distance d to compare the visual similarity of two patches is the *Sum of Square Differences* (SSD):

$$d_{SSD}(\Psi_p, \Psi_q) \;=\; \sum_{v \in (\mathcal{N}_p \cap (\mathcal{I}-\Omega))} \|\Psi_p(v) - \Psi_q(v+p-q)\|^2 \tag{1}$$

Note that other distances have been proposed in the literature such as the Bhattacharya distance [43] or the Hellinger distance [31] together with the SSD to compare patches probability density functions. Nevertheless the advantages of using these distances over the classical SSD is still unclear for the purpose of processing generic images.

B. Greedy approaches

The *greedy* approaches consist in gradually filling Ω with pixels or group of pixels until $|\Omega| = \varnothing$, each pixel of Ω being inpainted once. The sketch of these methods is expressed in 4 main steps:

1) Select one pixel p lying on $\delta\Omega$,
2) Search for the patch $\Psi_{\hat{p}}$ that best matches the patch Ψ_p centered on p,

3) Paste values from valid patch $\Psi_{\hat{p}}$ around p in Ω.
4) If $|\Omega| \neq \varnothing$, goto 1.

1) Criminisi Algorithm: The first two steps are of critical importance for the inpainting results. Naive implementations often leading to bad results, we focus now more in depth on these two steps that are essential parts of the Criminisi *et al.* [2] algorithm.

a) Filling Order: All three initial papers [26], [27], and [2] emphasize the importance of the filling order (i.e., how to select the pixel p on $\delta\Omega$), for this kind of inpainting techniques.

Filling in a layer fashion from the boundary of the initial mask to the center of it, in the so-called *onion-peel* way, is used in [26] and within a multiscale scheme in [27]. This filling order seems natural but in fact led often to unnatural results, especially near the center of the mask: important structures that should have been reconstructed first are lost.

To overcome this critical step, Criminisi *et al.* proposed in [2] a filling order based on known structures lying on $\delta\Omega$: a priority term P_p is computed for all pixels $p \in \delta\Omega$ as:

$$P_p = C_p \cdot D_p \tag{2}$$

where C_p is a measure of a *confidence*, and D_p is defined as a *data term* that takes care of the presence of structures in Ψ_p. The latter plays a critical role in the priority calculation and favors the continuation of structures that enter the mask Ω. Fig 1 (bottom) shows an inpainting at different iterations using Criminisi *et al.* algorithm and exhibits this main feature.

More precisely, the *confidence* term C_p can be seen as a measure of reliable information in the neighborhood of p. It is defined in [2] as:

$$C_p = \frac{\sum_{q \in (\mathcal{N}_p \cap (\mathcal{I}-\Omega))} C_q}{|\mathcal{N}_p|} \tag{3}$$

where $|\cdot|$ is the size of $\mathcal{N}_p$ (i.e., the number of pixels), and at the initialization, one sets:

$$\begin{cases} C_p = 1 & \forall p \in \mathcal{I} - \Omega \\ C_p = 0 & \forall p \in \Omega \end{cases}$$

This term has high values near the border of the initial mask, and decreases near the center of Ω. It tends then to inpaint first pixels having the most valid neighbors. This is a similar term that defines the *onion-peel* filling order in [26], [27].

The *data term* D_p reflects the local image structure around Ω and is defined in [2] as:

$$D_p = \frac{|\overrightarrow{\nabla I_p}^{\perp} \cdot \overrightarrow{n_p}|}{\alpha} \tag{4}$$

with $\overrightarrow{\nabla I_p}^{\perp}$ the isophote direction defined as:

$$\overrightarrow{\nabla I_p}^{\perp} = \left\{ \overrightarrow{\nabla I_q}^{\perp} \mid \underset{q \in (\mathcal{N}_p \cap (\mathcal{I}-\Omega))}{\arg\max} \|\overrightarrow{\nabla I_q}\| \right\}, \tag{5}$$

$\overrightarrow{n_p}$ is the normal vector to Ω at p, α a constant normalization factor (that can be in fact ignored since it is the same for all p), and $\overrightarrow{\nabla I_p}$ the color gradient vector at p. If several gradient vectors have the maximal norm, the one maximizing $|\overrightarrow{\nabla I_p}^{\perp} \cdot \overrightarrow{n_p}|$ is retained. This data term favors the reconstruction of local linear structure orthogonally crossing Ω at p, and is closely related to the notion of isophotes of *geometry-based* methods. As shown in Fig 1 (bottom), it strongly affects the filling-order and favors the continuation of structures that have been lost with a classical *onion-peel* scheme.

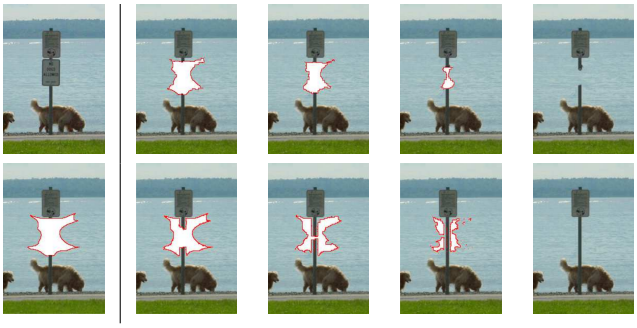
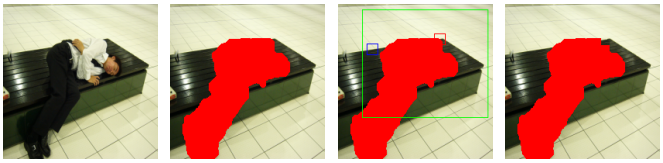


Fig. 1. Comparison between the *onion-peel* filling order (first row) and the *data aware* one (second row), courtesy of [2]. Thanks to the *data term*, the sign pole is correctly reconstructed.



(a) (b) (c) (d)

Fig. 2. Illustration of one iteration of the greedy approach [2]. (a) Original Image. (b) Masked image. (c) Search for the best patch. The target patch is depicted in red, the lookup area in green, and the best patch in blue. (d) Result after pasting the blue patch of (c).

Once all priorities $P_q, \forall q \in \delta\Omega$ have been computed, the pixel with the highest priority is chosen as the *target* pixel p to reconstruct.

b) Searching and matching patch and reconstruction: At each iteration, the best patch $\Psi_{\hat{p}}$ matching the non masked part of the patch Ψ_p centered on the target pixel p is searched all over the image. This area is actually often reduced to a square of smaller size centered at p in order to reduce the search space and then the overall processing time. More information about this key point are given at the end of this section.

Fig 2c shows for a given target pixel p, its patch Ψ_p (red square), the lookup region (green), and the best patch found $\Psi_{\hat{p}}$ found in it (blue square). Once the best patch $\Psi_{\hat{p}}$ has been found, its non masked part is drawn at p in $\mathcal{N}_p \cap \Omega$ (Fig 2d). Confidences are also duplicated from the confidence of Ψ_p:

$$C_q = C_p, \forall q \in \mathcal{N}_p \cap \Omega \tag{6}$$

2) Enhancements in the state-of-the-art: Many variations of this seminal algorithm have been proposed in the literature. Since the inpainting process is greedy, a minor change in the priority filling order could lead to significant changes in the final inpainted results. We illustrate this fact in Fig 3 by changing only the numerical gradient computation schemes (i.e., backward, centered and forward finite differences) in the *data term* computation, and keeping the other parameters of the algorithm constant. Significant changes of the results can

be observed. Moreover, by changing the patch size, completely different results are obtained, and in practice no numerical scheme seems better than the others. As a consequence, several works have then been conducted to produce enhanced priority terms.

• The authors of [29] proposed some heuristics to improve the priorities by using a tensor-based data term:

$$D_p = \alpha + (1-\alpha)\exp\left(\frac{\eta}{(\lambda_1-\lambda_2)^2}\right) \quad (7)$$

where λ_1 and λ_2 are the eigenvalues of the structure tensor [44], η and α are hyperparameters fixed at $\eta = 8$ and $\alpha = 0.01$. Using such tensors is intended to give a better modelization of the local image variations, and relies on two hyperparameters arbitrarily fixed by the authors.

• The authors of [30] proposed a sparse data term that measures for each pixel $p \in \delta\Omega$ the global similarity of a patch centered at p with patches contained in a user-defined neighborhood. The sparseness of a boundary pixel is computed as:

$$\rho_p = \|\overrightarrow{\omega_p}\|_2 \sqrt{\frac{|N_s(p)|}{|N(p)|}} \quad (8)$$

where $|N_s(p)|$ is the number of valid patches in the neighborhood of p (i.e., patches that do not contain masked pixels), $|N(p)|$ is the total number of patches in the neighborhood of p, and $\overrightarrow{w_p}$ is the vector containing the similarities w_{p,p_j} defined as:

$$w_{p,p_j} = \frac{1}{Z_p}\exp\left(\frac{-d(\Psi_p,\Psi_{p_j})}{\sigma^2}\right) \quad (9)$$

where $d(\cdot)$ measures the mean squared distance between the known parts of Ψ_p and a valid patch Ψ_{p_j}, Z_p is a normalization constant such that $\sum_{p_j \in N_s(p)} w_{p,p_j} = 1$, and σ is an hyperparameter fixed at $\sigma = 5$. This data term allows to discriminate between pixels of a texture from pixels belonging to the edge between two objects. Both pixels could have a high gradient, but the latter (edge pixels) is more sparsely distributed among its neighbors (low similarity), and is chosen first.

Fig. 3. Inpainting results with the Criminisi *et al.* algorithm [2] according to different numerical schemes for the gradient computation and different patch sizes. From left to right columns: Backward finite differences, centered finite differences, and forward finite differences. Patch size equal to 7×7 for the first row, 13×13 for the second.

• Recently, the authors of [31] proposed to amplify the data term in a nonlinear fashion to compute the priorities:

$$\hat{D}_p = \exp\left(\frac{D_p}{2\sigma^2}\right) \quad (10)$$

with σ an additional parameter. This variation enforces the data term in the presence of a strong isophote ($D_p \to 1$), and lowers it in flat areas ($D_p \to 0$).

We compare in Fig 4 these three data terms with the initial one proposed by Criminisi *et al.* [2] (with a forward finite differences scheme for the gradient computation). We manually tuned the additional parameters (the size of the neighborhood of Xu *et al.* method [30], and the standard deviation σ of the exponential data term (Eq. 10) of Martinez *et al.* method [31]) to get the best results. The remaining parameters (patch size and lookup size area) are kept constant. The pros and the cons of these priorities proposal together with our proposed data term will be discussed in Section IV-A.

Semi-supervised algorithms based on Criminisi *et al.* work have also been proposed. For instance [28] constrains the search space of patches by requiring the user to specify coarse search locations. The approach of [45] synthesizes image patches along user-specified curves in the unknown region using patches selected around the curves in the known region. The results of these semi-supervised algorithms highly depend on the manual interactions of the user, and are then difficult to evaluate, and indeed impossible to apply in a fully automatic way.

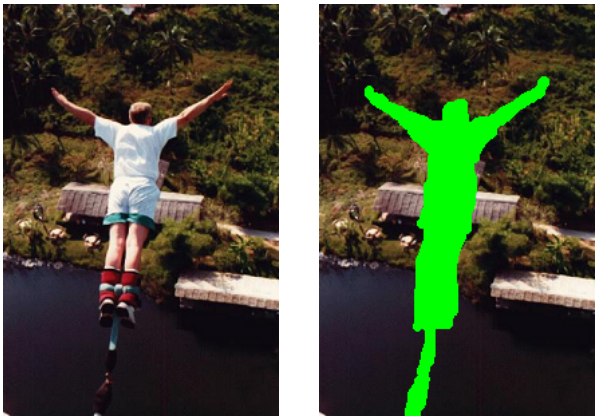
Several exemplar-based [29], [31], [46], [47] and other hybrid/global approaches [36], [42], [48] (detailed in the Section III-C and III-D) synthesize the missing part by mixing several patches. More precisely, the K best patches that match the target patch are found, and the missing part is synthesized pixel-wise with a linear combination of these patches. This kind of synthesis appears appealing but often produces blurry results, especially for textured regions.

C. Hybrid methods

Geometry-based methods have proven to reconstruct global structures correctly, but often fail at reconstructing textures [6], [7], [12]. On the other hand, greedy approaches are suitable for filling-in texture parts of an image, but have more difficulties to reconstruct missing structures globally. This is especially the case when the missing part of a structure is not present in the rest of the image, when the mask overlay a curved isophote for instance.

Since natural images often contain structures and textures at the same time, *hybrid* approaches combining both *geometry-based* and *exemplar-based* methods have been naturally explored in the literature.

Two main classes of hybrid methods have been proposed. The first one [32], [33] starts to decompose the image



(a) Original. (b) Mask.

(c) With D_p from [2] *(20.01s)*. (d) With D_p from [29] *(22.63s)*.

(e) With D_p from [30] *(50.5s)*. (f) With D_p from [31] *(19.90s)*.

Fig. 4. Comparison of inpainting results with different data terms with a constant patch size of 11×11 for all methods. The synthesis is the same for all the methods, and additional parameters for the methods of [30] and [31] have been manually tuned to get the best result. The processing times in parentheses stand for the whole inpainting process. The inpainting result with the same parameters and our proposed data term (see Section IV-A) can be seen in Fig.7 (bottom right).

to inpaint into structure and texture parts, leading to two sub-images that are inpainted independently with different approaches (geometry-based methods for the structure part, texture-synthesis for the textured part). These images are then recombined to reconstruct the final inpainted result. By separating the structures from the textures of an image, the first class of hybrid algorithms alleviates the potential drawbacks of geometry-based and texture synthesis methods, and these sub-images should become well suited for each inpainting step. This separation being a crucial step of such approaches, it is tackled in [32] with the use of a total variation minimization [49] and oscillating functions [50]. The authors of [33] decompose the image with a sparse prior onto a well suited dictionary, composed of two different sub-dictionaries to handle properly both structure and texture components of the image. The second class of hybrid method [34], [35] first tries to reconstruct strong edges of objects that enter the mask. This initial sketch serves then to define the lookup areas to further fill in the missing pixels with and exemplar-based approach. By continuing the main structures that are *broken* by the mask, the second class of hybrid approaches can reconstruct structures that are not present in the non masked part of the image (on which classical exemplar-based method can fail). While the method proposed in [34] uses tensor voting to infer the missing part of curves, level lines with Euler spiral are used in [35] to complete such curves. Once restored sketchs have been obtained, masked pixels are filled-in with a texture synthesis approach, while completed curves are used to constrain the search.

Hybrid approaches seems appealing but are in fact very difficult to apply in practice. Separating structure from texture parts of an image is indeed an open problem, which makes the task difficult especially in presence of both tiny and macro textures (which is often the case in natural images). Similarly, continuing main structures first before synthesizing missing pixels requires a (coarse) segmentation which can not be sufficiently general to handle numerous natural images cases. In addition, the high computation time of such kind of methods reduces their potential toward a practical use.

D. Energy-based methods

The last class of inpainting algorithms makes use of patches for the image analysis but synthesizes the missing part of the image pixelwise. They perform the reconstruction of the missing part via a minimization of a *global* energy (coherence of the reconstruction). They often require several iterations of the inpainting process to refine the reconstruction of the missing part, and sometimes in a multi-scale fashion.

Wexler *et al.* [36] propose a pixel-based synthesis as a weighted mean of patches that contain a pixel to be filled in. To ensure a global coherence, they propose to minimize the following measure of coherence, that measures for each missing pixel, the similarity between its surrounding patches and patches in the known part of the image:

$$Coh \;=\; \prod_{q\in\Omega} \max_{p\in\overline{\Omega}} sim(\Psi_p, \Psi_q) \qquad (11)$$

where $sim(\cdot)$ is a similarity measure between two patches based on a Gaussian function. They achieve this reconstruction in a pyramidal fashion (from lowest to full resolution) via an Expectation-Maximization optimization scheme. As noted in

[42], the weighted mean tends to blur uniformly reconstructed parts of the image, which can be annoying for tiny textures but at the same time performing this kind of blending can be useful to hide minor reconstruction artifacts. Together with the PatchMatch algorithm [51], it is actually the main component of the "Content-Aware Fill" tool in Photoshop CS5[1]. Mansfield *et al.* [39] further enhance the synthesis of Wexler's method by allowing various patch transformations (translation, rotation, scaling, or brightness shifts). Kawai *et al.* [48] also based their proposal upon Wexler's framework. Brightness change is, on one hand, taken into account in the search space such that the number of candidate patches is largely augmented, and on the other hand, the spatial locality of texture patterns is considered as an implicit constraint.

Komodakis *et al.* [37] cast the inpainting problem into a labeling problem of a MRF. An objective function is defined and optimized via a Priority Belief Propagation (Priority-BP) carrying improvements over standard BP to handle the important number of labels. The extension compared to state-of-the-art methods using optimization is that their method also tackles texture synthesis problems. Inpainting reconstructions with this method seem natural, but the computation time (up to 2 minutes for 256x127 images) makes this method hard to use in practice.

Pritch *et al.* [38] also cast the inpainting problem (and in fact many other such as image retargeting or object rearrangement) as a graph labeling problem. Here, the labels attached to the vertices of the graph are the relative shift of every pixel of the output image, and the optimal *shift-map* is computed via the minimization of an energy composed of two terms: a data term indicating constraints on the pixels, and a smoothness term that minimizes discontinuities between objects due to the discontinuities in the shift-map. In this method, the shift-map computation (pre-inpainting task) takes up to 30 seconds for 1600x1600 images.

He *et al.* [41] propose to compute principal patch offsets in the known part of the image, and to use them into a Photomontage framework [52]. The input image is shifted several times (by the principal offsets), then these images are stacked and the best cuts are found via the graph-cut algorithm. Poisson seamless blending technique [53] is further employed to minimize discontinuities between pasted image parts. This method provides a new way to perform image analysis for image inpainting. However the number of the offsets to extract (the parameters K [41]) really depends on the image topology and can lead to texture repetitions or local inconsistencies.

Finally Arias *et al.* [40] propose a general variational framework to handle both local and non-local (i.e., patches) methods. Several schemes are then derived via the selection of the appropriate patch similarity criterion: *patch NL-means*, *-medians*, *-Poisson*, and *-gradient medians* corresponding to similarity criterions based on L^2- and L^1- norms between patches or their gradients. The objective function adds to the similarity term (similar to equation 11) a term that measures the entropy of the similarities: for each pixel, the similarities are defined as a function that gives the probabilities to match any other pixel of the image. For a given pixel, the entropy of these similarities gives then the reliability of its matching pixel. A coordinate descent algorithm is used to minimize the objective function, alternating between similarity weights and the image updates.

This kind of approach requires several iterations to converge.

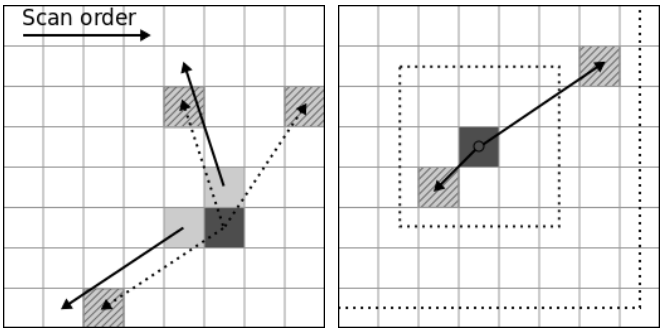
[1]http://www.photoshopessentials.com/photo-editing/content-aware-fill-cs5/

E. Source patch(es) search

Beyond the distance metric used to compute the similarity between patches, a key component both in term of execution speed and visual coherence of an inpainting result is the way to search for a good source patch for the reconstruction.

First, the chosen search method for a patch-based inpainting algorithm must be effective in term of reconstruction consistency, i.e., guaranteeing a sufficient quality for the final result. It must be fast as well, for the inpainting algorithm to be easily used in image editing applications. Seeking for the best source patch is often in fact the most costly part of an inpainting algorithm. A fast search often means approximate solution, leading locally to a worse quality of result than with a complete search. Satisfying the constraints of both speed and quality is so a quite difficult task in image inpainting context.

A simple and, in fact, efficient scheme to search for a source patch is to use a window of a defined size around the target patch. The search space being considerably reduced, far fewer candidates are checked during the seeking process than with a full search over the entire image. Such a full search is equivalent to computing an exact Nearest Neighbor Field (NNF) with respect to a given distance metric.



(a) Propagation. Plain arrows are the current patch offsets. The dotted arrows are the offsets that are checked during the propagation.

(b) Random search. The dotted squares represent the successive windows used to search a good approximate neighbor.

Fig. 5. Illustration of the two main phases of the PatchMatch algorithm: propagation (a) and random search (b).

Another scheme is to use a method for computing an Approximate Nearest Neighbor Field (ANNF) that helps to find a good source patch all over the image. Such an ANNF does not give the exact solution (i.e., the best source patch for all target patches) but an acceptable one. An ANNF between two images I_a and I_b is an offset map $\Phi(p)$ that provides for each $p \in \mathcal{I}_a$ the relative location of the center $q \in \mathcal{I}_b$ of

the patch Ψ_q that is the approximate nearest neighbor of Ψ_p. For the particular case of inpainting, I_a and I_b represent the same image, and are restricted to $\bar{\Omega}$. Several methods exist to compute such an ANNF:

- KD-trees have been widely used to retrieve an ANNF in a faster way than with a full search, especially for inpainting requiring several iterations [36]. KD-trees have further been improved [54] to accelerate the retrieval of approximate nearest neighbors by using a so-called propagation assisted technique. Note that this search method is used to efficiently compute statistics of patch offsets in [41].
- In 2009, Barnes *et al.* propose the PatchMatch algorithm [51] which is an iterative algorithm. It is based on three key observations:
  1) The dimensionality of offsets space is much smaller than those of patch space.
  2) The structures in natural images are often organized in a contiguous way.
  3) With the law of large numbers, "some nontrivial fraction of a large field of random assignments will likely be good guesses".

The ANNF $\Phi(p)$ is initialized randomly, and refined iteratively in two phases: propagation and random search. The propagation phase, reminiscent of the texture synthesis method [25], improves a link of the ANNF by looking to its neighbor links (see Fig 5). The second phase, named *random search* seeks randomly a better source patch within a window of decreasing radius. The goal of this second phase is possibly to escape from a local minima of the ANNF, and to insure a more global consistency.

These algorithms are interesting owing to the fact that they bring a global coherence with the global search, but also a local coherence thanks to information propagation. On the other hand, they were originally for *complete* images that are not subject to change after the ANNF computation. If they can be used as is, for image denoising or object detection, these methods require some adaptations in case of patch-based inpainting. First, they must be able to handle patches where information is missing. Then, as the ANNF becomes out of date when the image information change, an extra update phase is needed for each patch that is reconstructed.

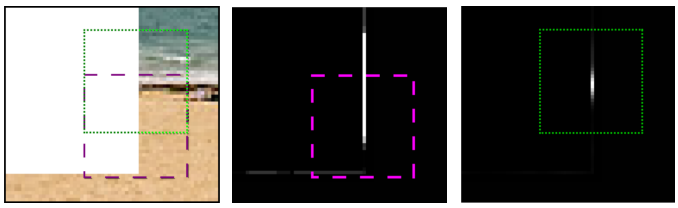
IV. Three improvements of exemplar-based Inpainting

The technical review of the state-of-the-art methods we made in the previous section has enlighted several enhancements to the key points of exemplar-based inpainting algorithm. But as we will discuss further, in some cases they are not the best suited solution. From our observations detailed above, we describe and discuss in this section the improvements we propose for each key point of the algorithm, and provide results and comparisons with the state-of-the-art methods to show their effectiveness in challenging cases.

A. Tensor-based data term

We showed that the priority term is one of the key features introduced in [2] for selecting an ideal reconstruction location at a given iteration of the inpainting algorithm. More particularly, the sub-priority term D_p (Eq. 4) will favor target points that are located on an image contour, oriented along the normal to the inpainting mask. Giving a high priority to these points is indeed a great idea, as it will naturally allow the important (contrasted) structures to be reconstructed and extended first.

Unfortunately, the original expression proposed for D_p in [2] has one major flaw: it assumes that the gradient $\overrightarrow{\nabla I_p}$ of the target point is computed as the maximum value of the image gradient in the non-masked neighborhood $\mathcal{N}_p \cap I$ of p (Eq. 5). In fact, when the fixed patch size N set for the reconstruction is large, the data term D_p will be high not only on the exact location of the image contour to extend, but also for every pixels $\in \delta\Omega$ whose distance from the image contour is less or equal than N (Eq. 5). This dilation effect of the data term gives usually too much importance to these neighboring target candidates, and they can be unfortunately selected instead of the exact contour point, particularly when the mask shape makes these points best candidates regarding the confidence term C_p that is the other sub-priority term. This fact is illustrated on Fig.6a on a real case. The data term (Eq. 4) is high for all target points around the frontier between the sand and the see, and the final selected target patch to reconstruct will not be centered at this interface (Fig.6b). A patch containing only sand is likely to be pasted there, and will break the sand-see edge.



(a) Local inpainting configuration. (b) Original D_p. (c) Our modified D_p.

Fig. 6. Illustration of the impact of the data terms. The dashed (b) and dotted (c) squares are the priority patches respectively chosen with the original D_p from [2] (Eq. 4) and with ours (Eq. 12).

As discussed in Section III-B, several data terms have been proposed in the literature. While they try to overcome the main issue of the initial data term, they suffer from different shortcomings that we discuss here.

The data term proposed in [29] is based on local structure tensors (Eq. 7). Using such tensors is a good idea since tensors modelize the local image variations and are more geometry-aware. Nevertheless, this data term (Eq. 7) relies on ad-hoc hyperparameters η and α whose role is unclear. More annoying, as defined in [29], it does not take into account the normal vector to Ω. Hence a contour that is tangent to Ω will have a strong anisotropy, and can led to high priorities for its pixels, which is not a desirable property.

The sparse-based data term proposed in [30] mainly allows to efficiently deal with texture pixels lying on $\delta\Omega$. With the Criminisi *et al.* data term (Eq. 4), most of these pixels can have high gradients, and then high priority values. Since a patch

centered on such a pixel is redundant over its neighborhood, the data term proposed by Xu *et al.* (Eq. 8) lowers its priority. Nevertheless, this data term has two drawbacks: first, the neighborhood of a pixel p is a square area centered at p whose size has an important impact on the sparseness term. In fact it depends of the *size* of the texture patterns, which is not predictable. Second, this data term is cumbersome in term of processing times. At the first iteration, for each pixel of the boundary, one has to compute several SSD distances between patches which is *very* time consuming. Although this has to be done at next iterations for only a small number of pixels (those who have been pasted at the previous iteration), the processing times of the whole process is one to two orders of magnitude higher than for the other data terms.

Finally, the method proposed in [31] enforces the data term in an exponential fashion for the computation of the priorities (Eq. 10). The rationale behind this amplification is to enforce much more the data term in the presence of a strong isophote ($D_p \rightarrow 1$), while not amplifying it in flat areas ($D_p \rightarrow 0$). The main issue of this heuristic is, as for the sparse-based data term described above, the presence of an additional parameter σ. This extra parameter has in fact a strong influence, and *in fine* has to be manually tuned, which can be unpractical.

To overcome these issues, we propose a geometry-aware approach using structure tensors [44]. They are considered to modelize image variations inside a candidate patch. First, this has the interest of estimating the local image structures using a channel-correlated approach (correlation between the R, G, B channels will be taken into account for considering the local geometry of color images). Second, we take advantage of the algebraic properties of the tensor sum, to allow target patches containing structures with multiple orientations (typically textures) to score favorably regarding the local orientation $\overrightarrow{n_p}$ of the mask normal at p. Our proposed data term D_p is then:

$$D_p = \|\mathbf{G}_p \, \overrightarrow{n_p}\| \tag{12}$$

where $\mathbf{G}$ is a weighted average of structure tensors estimated on non-masked parts of the target patch Ψ_p:

$$\mathbf{G}_p = \sum_{q \in (\mathcal{N}_p \cap (I-\Omega))} w_q \, \overrightarrow{\nabla I_q} \overrightarrow{\nabla I_q}^T$$

and w is a normalized 2d Gaussian function centered at p. This new data term can be understood as follows.

- When $\mathbf{G}$ is strongly anisotropic (i.e., $\mathbf{G} \approx \mathbf{u}\mathbf{u}^T$), there is one clear single image contour inside the target patch Ψ_p, oriented along $\overrightarrow{\mathbf{u}}^\perp$. So, our data term (Eq. 12) becomes approximatively equal to $| < \overrightarrow{\mathbf{u}}, \overrightarrow{n_p} > |$ which will be high when the contour is oriented along the normal of Ω. Note also that D_p will be higher for the target points p that are located precisely on the image contour (as the Gaussian weights w favor the centering of the target patch on the contour itself). There are no more dilation effects that can cause target points around the actual contour to be selected in priority.
- When $\mathbf{G}$ is isotropic (i.e., $\mathbf{G} \approx \lambda \; Id$) with small values (i.e., $\lambda \approx 0$), there are very few variations inside Ψ_p and we are located on an homogeneous region. In that case, our data term (Eq. 12) is low, as it is roughly equal to λ.
- When $\mathbf{G}$ is isotropic with high values (i.e., $\lambda >> 0$), the target patch Ψ_p contains a lot of variations oriented in different directions (contrasted and complex structure). In this case, our data term (Eq. 12) is always high whatever the orientation of the mask normal $\overrightarrow{n_p}$ is.

The interest of this priority term replacement is illustrated on Fig.6c where we can clearly see the better localization of the prioritized target points, as well as the correctness of the selected patch (the one with the highest priority) for the reconstruction at a given iteration. As inpainting is an iterative process where each patch selection depends on the previous iteration, we have observed this has indeed a dramatic incidence on the quality of the reconstructed image (see Figures 4 and 7 for instance).

Fig. 7. Illustration of the proposed *data term* for the *Bungee jumping*. Top: the original image and the part we want to remove. Middle: data terms of Criminisi *et al.* [2] (left) and ours (right) with a heat color map (dark red for low data term values, and yellow for high values). Bottom: resulting inpainted images. Due to the dilation effect of the original data term (Eq. 4), the roof of the shelter is broken (left). One can see also some *texture garbage* on the water. The proposed data term (Eq. 12) concentrates on the interface between the roof of the shelter and the jumper and avoids the copy of a *forest-only* patch onto the roof. Similar results with state-of-the-art data terms can be found in Fig.4.

B. Proposed lookup strategy

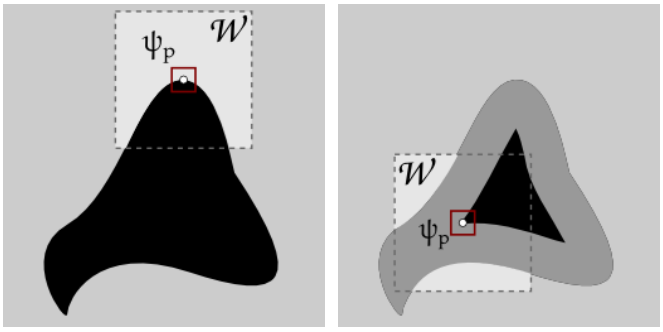
In Criminisi-like exemplar-based inpainting state-of-the-art, many methods try to improve the *priority term* P_p, often by modifying C_p or D_p, or trying to improve the way to compare patches, The way to search a good patch for the reconstruction often stays unclear while it is an important key point for this kind of algorithm to work well. For our experiments we tried several methods to search good candidates: a search window, (an adaptation of the) PatchMatch algorithm, and offset statistics.

A search window can be fast, but is tricky to use in case of large holes. Also, because of the spatial proximity of the samples according to the target patch, it does not insure enough global consistency. PatchMatch algorithm needs many adaptations to be used in a Criminisi-like exemplar-based inpainting. The SSD formula to use must take the hole in account, and the ANNF must be updated each time a patch is reconstructed. The latter makes in fact the PatchMatch algorithm far slower than its original version [51]. Finally, within for offsets statistics framework [41], as the number of selected histogram peaks is small, some local specific cases are left out, leading to local inconsistencies. Our contribution combine the speed of a search window, the global consistency of PatchMatch, and the speed and local consistency of offset statistics. The idea is to search inside a window, and keep each target patch offset to use them further in the algorithm. This way we build a set of search sites where we perform a fast windowed search.

1) Lookup state-of-the-art methods analysis:

a) Search window: A search window is an area of a much smaller size than the image one, where a good source patch is sought during inpainting process. This way, only a few patches are scanned during the search, and the reconstruction is then considerably accelerated since only a few SSD are computed. On the other hand, the use of a search window can be problematic. First, the fact that the window must be small enough size to accelerate noticeably the process may lead to poor reconstructions. Patterns and textures are more likely to be repeated in the final result. Note that a too large size window may also lead to poor reconstructions since too many patch candidates may be considered for the reconstruction and finally lowers the global consistency of the inpainted part.

An important problem in such a search scheme, is that the patches are searched only inside $\mathcal{I}\backslash\Omega$ to avoid the copy of synthesized pixels. When the mask size is larger than the search window, everything goes fine for the first iterations (Fig. 8a). But, after some iterations, the search space becomes too small to find a good candidate for the reconstruction (Fig. 8b). An ad-hoc solution [26] is to enlarge the search window to increase the number of candidates to insure finding a good one, at the cost of gradually longer iterations. Even if the use of a search window induces local coherence in the reconstruction (small SSD), which is a real benefit in patch-based inpainting, it lacks flexibility in the choice of the parameters and provides no guarantee of global coherence: a local coherence does not imply a global one.



(a) Inpainting at iteration 0. (b) Inpainting at iteration n.

Fig. 8. Illustration of the window size problem. Much less samples can be checked in the search window at iteration n.

b) PatchMatch: PatchMatch algorithm [51] was originally described as an algorithm to map patches from an image to patches of another one. Therefore, using this algorithm for exemplar-based image inpainting requires some adaptations. In state-of-the-art inpainting methods using PatchMatch, the way to handle missing data often stay unclear despite the importance of this key point. As the image hole color can corrupt the PatchMatch result, the missing data must be handled properly. The most obvious way to do so is to change the way to compare two patches. In the case of exemplar-based inpainting, Eq. 1 can be used for example. The second adaptation is related to the exemplar-based inpainting process. PatchMatch is generally performed on a still image, but in the case of image inpainting, the image content changes during the process. Therefore, $\Phi(p)$ has to be updated for all the centers of the patches whose content has changed, i.e $\{q \mid \mathcal{N}_q \cap \mathcal{N}_p \neq \varnothing\}$. This phase is very time consuming in practice and makes PatchMatch quite difficult to use in our case.

c) Offset statistics: An interesting idea with an ANNF is to benefit from offsets statistics [41], and it can easily be adapted to exemplar-based inpainting. We performed experiments where a pre-inpainting phase serves as a basis for a plain computation of the ANNF (using PatchMatch).

The K prevailing offsets can then be extracted to guide the search of patch candidates within the second inpainting pass. This scheme works fine in some cases by providing natural inpainted results. Nevertheless, by using only the K prevailing offsets, specific cases are left out and this may lead to unnatural local reconstructions. Some offsets can be artificially created by manually drawing strokes on the image (Fig. 18 in [41]) to partially solve the problem. In this case, this method becomes a semi-automatic method and goes beyond the scope a fully automatic framework we aim at.

2) Fast and smart search contribution: The contribution we propose gather many ideas from the previously exposed search algorithms. The proposed strategy is first to search the nearby reconstructed target patch offsets $\Phi(p)$, and then use them as several new search *sites* to find a good source patch for the reconstruction (Fig. 9). The search area then consists

in several search windows, each centered at the location of the center of the previous source patch (w.r.t. $\Phi(p)$). In practice, the set of nearby reconstructed patches $\{\Psi_c\}$ of a target patch Ψ_t are those for which $\mathcal{N}_c \cap \mathcal{N}_t \neq \varnothing$.

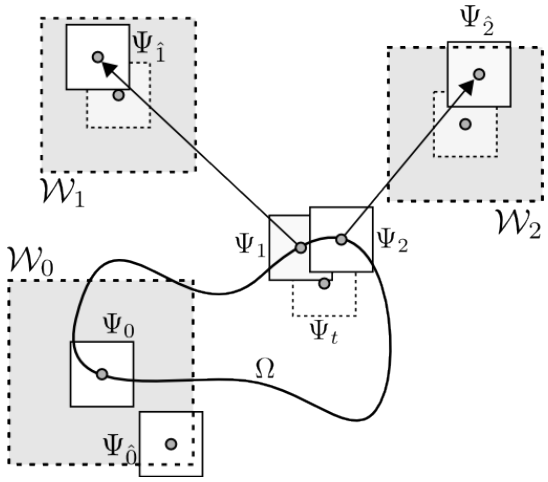


Fig. 9. Illustration of our fast and smart search for exemplar-based inpainting algorithms. Each reconstructed patch Ψ_1, Ψ_2 provides respectively a site $\mathcal{W}_1, \mathcal{W}_2$ to help finding a good candidate to reconstruct Ψ_t. Note that a site $\mathcal{W}_t$ is also centered on Ψ_t (not shown here for visualization purposes). As Ψ_0 has no nearby reconstructed patch, its search window $\mathcal{W}_0$ is simply centered on it.

For each Ψ_p to be reconstructed, the search of a good candidate is performed as follows. First, the set of offsets $\Phi(p)$ of patches that have already been reconstructed are sought inside a window of the size of Ψ_p. Note that the offset $(0,0)$ is also taken into account. The size of the look-up windows $\mathcal{W}(p)$ is then computed with the following formula:

$$w_{size}(p) = \begin{cases} \gamma & \text{if } |\Phi(p)| = 1 \\ \gamma\alpha\sqrt{|\Phi(p)|} & \text{otherwise} \end{cases} \quad (13)$$

where γ is the maximum size of the search window, and α is a scale factor giving the amount of space of the main window to grant to the search sites: the more sites are found (and further visited), the smaller they are. Once these search sites have been defined, a classical window search is performed into these multiples sub-windows:

$$\Psi_{\hat{p}} = \{\Psi_q \mid q = \underset{\hat{q} \in \Phi(p)}{\arg\min} \underset{\cup}{} d(\Psi_p, \Psi_{\hat{q}})\}$$

The advantages of this proposed search scheme are multiple:
• As search sub-windows often overlap themselves in practice (but not systematically), the number of candidates is always lower than the one within the initial search window strategy [2]. This increases the speed of the search phase by a non-neglictible factor.
• The window size problem illustrated in Fig. 8 does not occur anymore, since the search sites are always defined to be outside the inpainting mask.
• As fewer candidates are investigated, the aproximate nearest neighbor may not be optimal (in the sense of the SSD), but often results visually better. Search sub-windows are in fact centered at geometrically coherent locations according to the previous inpainting iterations (i.e., previously pasted patches). As a consequence, the best patch may not be the optimal for local coherence (in terms of SSD), but it provides a more global consistency to the results.
• Reminiscent of the method of [25] for texture synthesis, our search strategy often achieves a better global consistency by searching only at smart locations, and then becomes less sensitive to the patch size. One can copy/paste larger structures and textures and have a correct reconstruction of these with smaller patch sizes than with a classical search.

Figure 10 illustrates our proposed search scheme and compares it to the classical window search on a difficult image that contains both structures and textures. Although inpainting results looks similar, inpainting with our method requires only 17×17 patches (bottom left) when 23×23 patches are required with the classical window search (bottom right). Inpainting with the classical window search and too small patches (17×17) degrades too much the structure of the columns (top right).

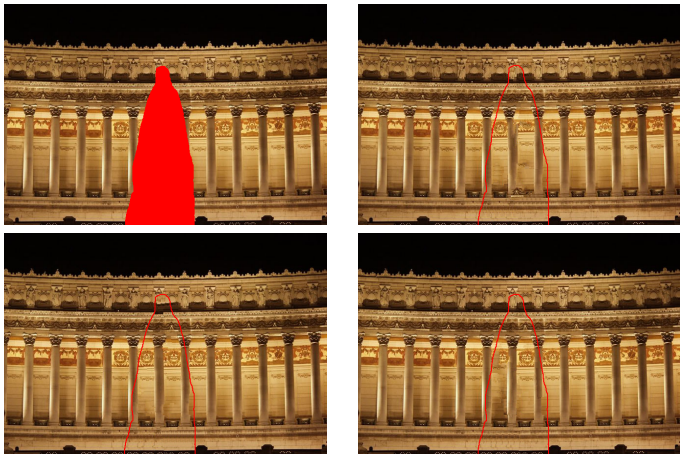


Fig. 10. Comparison of inpainting results with the classical window search (right column) and our search scheme (bottom left). The best results have been obtained with 23×23 patches for the classical window search (bottom right) and with 17×17 patches for our search scheme (bottom left). The initial window size is found to be equal for both (400×400). Inpainting with the classical window search and 17×17 patches (top right) degrades too much the structure of the inpainted columns.

Gathering ideas from several state-of-the-art algorithms, we succeeded in creating an original search algorithm that is able to maintain both a certain local and global consistency over the image reconstruction, and this with a quite high speed. But, as for pure exemplar-based inpainting algorithm, some seams might appear in the final result. In the next section we describe the way we found to strongly reduce these possible artifacts.

C. Spatial patch blending

Criminisi-like exemplar-based inpainting algorithms often produce block effect artefacts during the reconstruction. This is mainly caused by the copy and paste of patches chunks that do not match perfectly on their common boundaries, even with a smart selection scheme. Some methods are used to try to

reduce these artifacts. In [29], a K-means algorithm is used, but depending on the number of patches K, the result can look very blurry. The pixel-based reconstruction method of [36] provides an intrinsic way to not create artefacts, but the results can also be blurry, or present some texture warping. Reminiscent of the the Poisson blending method [53] used in [28], we proposed in [3] a fast way to reduce these block effect artifacts by spatially blend nearby patches one another where artifacts are located. Contrary to [28] where the interfaces between blocks are known and fixed in advance (since patches are fully copied), artifacts are first detected by analyzing the result and the reconstructed patch offsets provided by the inpainting process. Then, the spatial blending algorithm is applied on the inpainting result to reduce the visibility of the artifacts.

1) Artefact detection: The results provided by a exemplar-based inpainting algorithm do not present artefact everywhere. Our detection method for the artifacts is empirically based on the two following complementary hypotheses: 1) there are sharp variations of luminosity or color where artefacts are located, and 2) patches stuck side by side and coming from very different locations are most likely to be of different natures. The pipeline of the detection process (Fig. 11) is divided on two parts: break field computation, and break point extraction.

a) Break field: The break field associates each points of an image, the strength of the artifact that is locally present (if any). It is defined by the two above hypotheses, and modeled with the following equation:

$$R(p) = \frac{|\nabla I_p| \ . \ |\nabla \Phi(p)|}{\alpha} \quad \text{with} \quad R(p) \in [0,1] \tag{14}$$

where $\alpha = \max_{q \in \mathcal{I}}(|\nabla I_q| \times |\nabla \phi(q)|)$ is a normalization factor. This equation makes the break field higher when both sharp variations are presents (high $|\nabla I_p|$) and patches come from different locations (high $|\nabla \Phi(p)|$). On the contrary, the break field strength gets locally lower when one of the two above key conditions is not respected.

b) Break points: In order to blend the inpainting result only where needed, the break point set $\mathcal{E}$, points of the highest break strength, are selected by using a threshold value τ. The higher τ, the less blending strength. Thanks to the normalization factor α used in equation 14, the values of τ are in $[0,1]$. It makes these parameters easy to use.

2) Spatial patch blending algorithm: The principle of the spatial patch blending is to merge overlapping parts of nearby patches together in a way that the boundaries of these patches become much less visible, and even invisible. For each $p \in \Omega$, the spatial blending performs a linear combination of all the pixels overlapping at p from the set of reconstructed patches $\{\Psi_1, \ldots, \Psi_n\}$. In the following, we detail the core of our *tensor-based* patch blending that has the advantage of blending/removing incoherent patch data while preserving the significant structures and textures as much as possible.

a) Tensor model for spatial patch blending: We propose a tensor-based spatial blending algorithm that is much more careful about local image structures and textures than our previously proposed algorithm [3].

While in [3] the blending for a given pixel was performed equally in all direction (isotropic model), we propose here to model into a unified model the following three properties: (1) blend a flat area with a strong strengh in all direction (isotropy with a high amplitude), (2) blend a textured area with a small strengh in all directions (isotropy with a small amplitude), and (3) blend a structured area with a weak strengh in the structure direction (anisotropy).

The first step is then to construct tensors that fit this model, the so-called *blending-tensors* $\mathbf{B}$. The eigen values $\lambda_{\mathbf{B}\{1,2\}}$ and eigen vectors $\mathbf{e}_{\mathbf{B}\{1,2\}}$ of blending tensors $\lambda_{\mathbf{B}}$ will represent respectively the bandwidth and the direction of the spatial patch blending to be applied locally. As structure tensors $\mathbf{S}$ already provide a good local geometry analysis for each pixel of the image, we propose to use structure tensors as a basis for the building our *blending tensor* model.

Constructing blending tensors is performed in three steps:

• The eigen values of structure tensors used as a basis are fully dependant of the image value range. The first step is then to normalize them:

$$\hat{\lambda}_{\mathbf{S}(p)i} = \frac{\lambda_{\mathbf{S}(p)}}{\max\limits_{p \in \mathcal{I}} \lambda_{\mathbf{S}(p)i}} \tag{15}$$

• As the local blending bandwidth (i.e., the eigen vales of the blending tensors) is defined by the ratio between the smallest and the biggest eigenvalue of the struture tensors, this step intends to modify eigen values $\hat{\lambda}_{\mathbf{S}(p)i}$ depending on this ratio in order to have new eigen values $\lambda_{\mathbf{B}i}$. Proposed function, inspired by partial differential equations for diffusion [12], is the following:

$$\lambda_{\mathbf{B}i} = \frac{1}{\left(1 + \hat{\lambda}_{\mathbf{S}1} + \hat{\lambda}_{\mathbf{S}2}\right)^{\gamma_i}} \tag{16}$$

where γ_i ($i \in \{1,2\}$) are parameters controlling overall tensor isotropy. Examples of the effect of different configurations of γ_i are provided in Fig. 12.

• This final step consists in building the final blending tensor for each pixel. It is defined as a positive symmetric and positive definite matrix and expressed as:

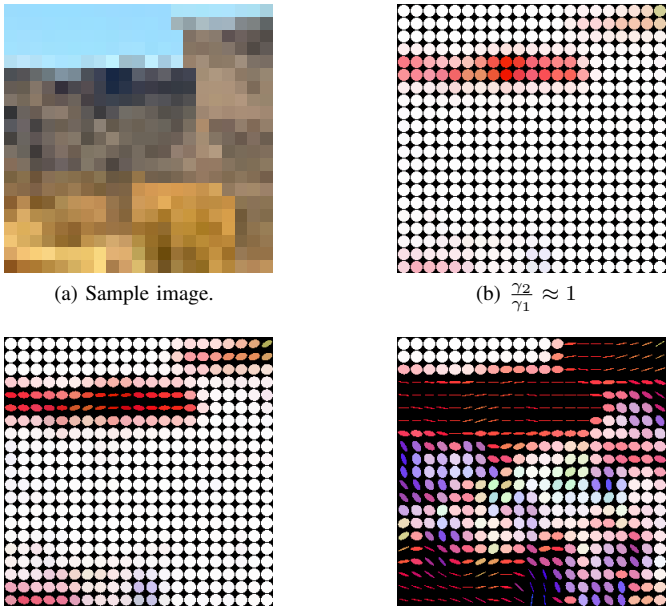
$$\mathbf{B} = \lambda_{\sigma\mathbf{B}1}\mathbf{e}^{\perp}_{\mathbf{S}1}.^T\mathbf{e}^{\perp}_{\mathbf{S}1} + \lambda_{\sigma\mathbf{B}2}\mathbf{e}^{\perp}_{\mathbf{S}2}.^T\mathbf{e}^{\perp}_{\mathbf{S}2} \tag{17}$$

where $\lambda_{\sigma\mathbf{B}i} = \sigma_{\mathbf{B}}\lambda_{\mathbf{B}i}$, are the bandwidth-scaled eigen values of the blending tensors.

b) Tensor-directed patch blending: The direct application of the patch blending process aims at blending the whole image in a pixel-wise fashion using the proposed tensor model $\mathbf{B}(p)$. The proposed approach computes a linear combination of pixels coming from several patches overlapping each other, with respect to the local geometry. For each $p \in \Omega$ in each image channel k, the blended result is computed as :

$$J^k(p) = \frac{\sum\limits_{\psi_q \in \Psi_p} w_{\mathbf{B}}(p,q) \ \psi^k_q(p-q)}{\varepsilon + \sum\limits_{\psi_q \in \Psi_p} w_{\mathbf{B}}(p,q)} \tag{18}$$

where $\Psi_p = \{\psi_0, \ldots, \psi_n\}$ is the set of source patches overlapping p, and $w_{\mathbf{B}}$ is an anisotropic Gaussian weight



(a) Sample image. (b) $\frac{\gamma_2}{\gamma_1} \approx 1$ (c) $\frac{\gamma_2}{\gamma_1} > 1$ (d) $\frac{\gamma_2}{\gamma_1} \to \infty$

Fig. 12. Illustration of the effect of the values of parameters γ_i.

function defined as follows:

$$w_{\mathbf{B}}(p,q) = e^{\frac{X^T \mathbf{B}(p)^{-1} X}{2\sigma_{\mathbf{B}}^2}} \quad \text{with} \quad X = q - p \qquad (19)$$

This blending scheme has the effect of blending strongly flat areas, while blending slightly textures or in the contour direction. In the first case, unwanted seams disappear while, in the second case, it preserves the sharpness of synthesized structures and textures.

c) Making blending faster: The inner formulation of our proposed tensor-based patch blending, as pointed in [3], is quite slow to process in practice since the tensors have to be defined for each pixel of the mask. A simple and effective way to make the blending faster is to approximate the geometry of the reconstructed patches using only a blending tensor on their center. This approximation makes sense since the center of a reconstructed patch is related to the processing order built upon the pixels anisotropy (see Section IV-A). While in the inner formulation (see above) there are as many weight functions as patches overlapping the position of a specified pixel to compute. In this faster formulation only one weight per patch has to be computed. This approximation does not degrade too much the blending result, while it makes it faster such it can easily be applied. The main parameter of our blending algorithm is the blending strength, which is in fact closely related to the patch size. Since the blending is fast and is somehow disconnected from the core of the inpainting process, it can easily be manually tuned for specific needs.

Fig. 13 shows an inpainted image, and compares our tensor-directed patch blending method with the one in [3]. Particularly, one can see the removal of artifacts while slightly better keeping the sharpness of the structures and textures.

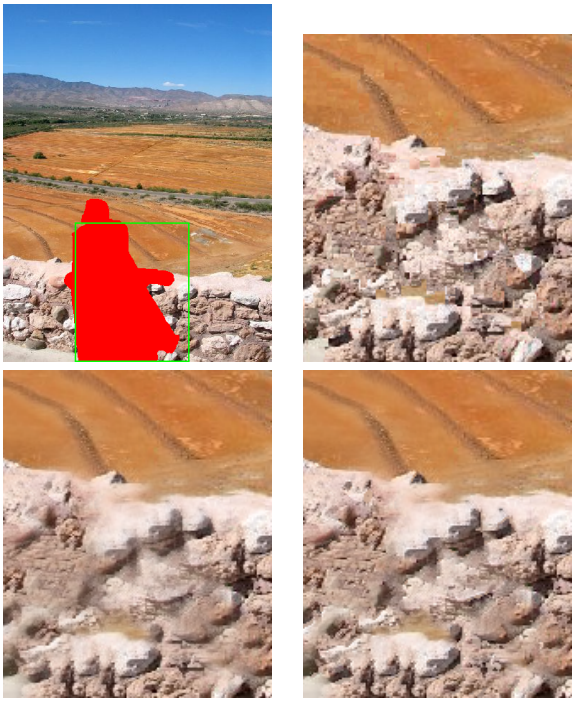


Fig. 13. Comparison of the proposed tensor-directed patch blending method (bottom right) and the one in [3] (bottom left).

V. Results

Fig. 14 illustrates our inpainting method on several natural images containing both structures and textures. Our results are exclusively compared to other state-of-the-art methods results, extracted from their respective papers, or computed from scratch when the corresponding code was made available. As one can notice, these qualitative comparisons exhibit visually equivalent or better final results on these difficult natural images.

Beyond not-well chosen parameters, our method can give unsatisfactory inpainting results on difficult images. Fig. 15 shows such bad results. For these images, the missing information can not be properly filled since it does not exist in the rest of the image. In fact, most of the exemplar-based methods can not deal with such cases, and this is the main limitations of such approaches (including ours).

The complexity of the whole algorithm can be detailed as follows:

- Similarly to the data terms of the literature (except the sparse-based one of [30]) detailed in Section III-B2, our tensor-based data term has a negligible complexity compared to the complexity of the whole algorithm. The slight differences of processing times shown in Figure 4 simply come from a different processing order that may induce more (or less) iterations.
- The complexity of our search scheme is difficult to compute since it relies on many parameters (patch size, lookup window size, ...). We then compare it to the classical window search scheme on the number of computed SSD required to inpaint an image. Figure 16 plots these values for all the images of this paper. Our search scheme requires between 2 to 5 times less

Algorithm 1 Complete inpainting algorithm.

{Inpainting part}

while $\|\Omega\| \neq \varnothing$ **do**

  $\forall p \in \delta\Omega$, compute priority $P_p = C_p \cdot D_p$ (Eq.2):

    $C_p = \frac{\sum_{q \in \mathcal{N}_p \cap (\mathcal{I}-\Omega)} C_q}{|\mathcal{N}_p|}$

    $D_p = \|\mathbf{G}_p \overrightarrow{n_p}\|$ (Eq.12)

  Select the target patch Ψ_p centered at p st.:

$$p = \arg\max_q P_q$$

  Construct search sites (see Section IV-B2) w.r.t. nearby reconstructed target patch offsets $\Phi(p)$ with size

$$w_{size}(p) = \begin{cases} \gamma & \text{if } |\Phi(p)| = 1 \\ \gamma\alpha\sqrt{|\Phi(p)|} & \text{otherwise} \end{cases}$$

  Search for the best patch $\Psi_{\hat{p}}$ (Eq.1)

  Paste $\Psi_{\hat{p}}$ in $\mathcal{N}_p \cap \Omega$

  Update confidences : $\forall q \in \mathcal{N}_p \cap \Omega, C_q = C_p$

  Update $\delta\Omega$ and associated confidences

  Update Φ

{Blending part: Fast method}

J: blending output

A: scalar image of dimension of I

for each reconstruction target p **do**

  $w_{\mathbf{B}}$: anisotropic weight function (Eq.19) based on blending tensor $\mathbf{B}(\hat{p})$ (Eq.17)

  $\tilde{\Psi}_{\hat{p}}$: Point-wise multiplication of $\Psi_{\hat{p}}$ by $w_{\mathbf{B}}$

  Draw $\tilde{\Psi}_{\hat{p}}$ at p in J

  Draw $w_{\mathbf{B}}$ at p in A

Point-wise division of J by A

$\forall p \in \Omega, I_p \leftarrow J_p$

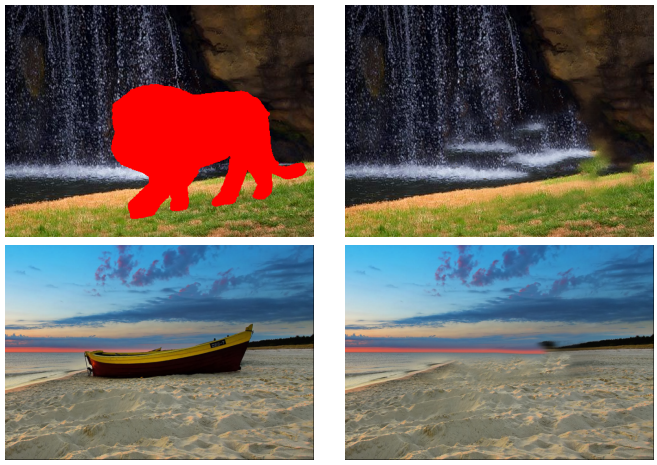


Fig. 15. Example of a failure case. Here, the missing information can not be properly filled since no patches lying at the interface of the three areas (water, grass, and stones for the *lion* image, and sand, sea, and sky for the *boat* image) can be found in the rest of the image.

SSD computation, which is known to be the main bottleneck of such approaches [51].

- The proposed approximation of our tensor-directed anisotropic blending is some somehow similar to the isotropic one previously proposed in [3]. The spatial blending (in its fast version) has a negligible cost in view of the whole process. Moreover, once the image is inpainted, the parameters of our blending algorithm can be tuned at interactive rate.

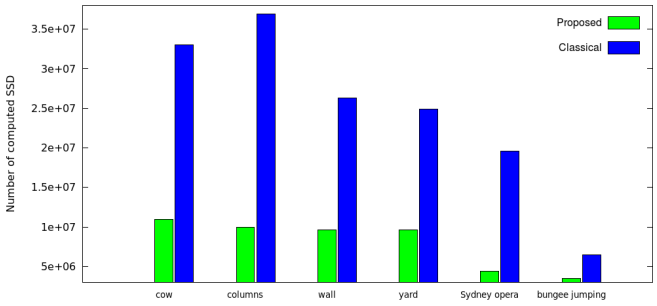


Fig. 16. Comparison of the number of SSD required to inpaint images of this paper with the classical window search and the proposed search scheme.

VI. Conclusion

In this paper we have presented a review of examplar-based image inpainting methods in a technical fashion, with a focus on the *greedy* ones. Each key point of the algorithm is then compared both technically and visually to similar state-of-the-art solutions. By using structure tensors, the data term of our method is robust and insures a global coherence in the reconstruction. Then, the strategy we proposed to search good reconstruction patch allows synthesizing micro and macro textures with a good local coherence. Finally, the method proposed to spatially blend the reconstructed patches in the final result beautifies and reduces typical block artifacts that are present.

Further work include the extension of our inpainting approach to video and stereo images/video inpainting.

REFERENCES

[1] C. Guillemot and O. Le Meur, "Image inpainting: Overview and recent advances," *Signal Processing Magazine, IEEE*, vol. 31, no. 1, pp. 127–144, 2014.

[2] A. Criminisi, P. Pérez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE T. Image Process.*, vol. 13, no. 9, pp. 1200–1212, Sep. 2004.

[3] M. Daisy, D. Tschumperlé, and O. Lézoray, "A fast spatial patch blending algorithm for artefact reduction in pattern-based image inpainting," in *SIGGRAPH Asia 2013 Technical Briefs*. ACM, 2013, pp. 8:1–8:4.

[4] M. Daisy, P. Buyssens, D. Tschumperlé, and O. Lézoray, "A smarter exemplar-based inpainting algorithm using local and global heuristics for more geometric coherence," in *ICIP*. IEEE, 2014, pp. 1–5.

[5] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *ACM T. Graphic.*, 2000, pp. 417–424.

[6] S. Masnou, "Disocclusion: a variational approach using level lines," *IEEE Trans. Image Process.*, vol. 11, no. 2, pp. 68–76, 2002.

[7] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera, "Filling-in by joint interpolation of vector fields and gray levels," *IEEE Trans. on Image Process.*, vol. 10, no. 8, pp. 1200–1211, 2001.

[8] T. F. Chan and J. Shen, "Nontexture inpainting by curvature-driven diffusions," *Journal of Visual Communication and Image Representation*, vol. 12, no. 4, pp. 436–449, 2001.

[9] J. H. Elder and R. M. Goldberg, "Image editing in the contour domain," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 3, pp. 291–296, 2001.

[10] J. Shen and T. F. Chan, "Mathematical models for local nontexture inpaintings," *SIAM J. Appl. Math.*, vol. 62, no. 3, pp. 1019–1043, 2002.

[11] S. Roth and M. J. Black, "Fields of experts: A framework for learning image priors," in *CVPR*, vol. 2. IEEE, 2005, pp. 860–867.

[12] D. Tschumperle and R. Deriche, "Vector-valued image regularization with pdes: A common framework for different applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 4, pp. 506–517, 2005.

[13] F. Bornemann and T. März, "Fast image inpainting based on coherence transport," *J. Math. Imaging Vis.*, vol. 28, no. 3, pp. 259–278, 2007.

[14] L. Demanet, B. Song, and T. Chan, "Image inpainting by correspondence maps: a deterministic approach," *Appl. Comput. Math.*, vol. 1100, pp. 217–50, 2003.

[15] A. Levin, A. Zomet, and Y. Weiss, "Learning how to inpaint from global image statistics," in *ICCV*. IEEE, 2003, pp. 305–312.

[16] M. Elad, J.-L. Starck, P. Querre, and D. L. Donoho, "Simultaneous cartoon and texture image inpainting using morphological component analysis (mca)," *Applied and Computational Harmonic Analysis*, vol. 19, no. 3, pp. 340–358, 2005.

[17] T. F. Chan, J. Shen, and H.-M. Zhou, "Total variation wavelet inpainting," *J. Math. Imaging Vis.*, vol. 25, no. 1, pp. 107–125, 2006.

[18] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *IEEE T. Image Process.*, vol. 17, no. 1, pp. 53–69, 2008.

[19] J.-F. Cai, R. H. Chan, and Z. Shen, "A framelet-based image inpainting algorithm," *Appl. Comput. Harmon. A.*, vol. 24, no. 2, pp. 131–149, 2008.

[20] A. Buades, B. Coll, and J.-M. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Model. Sim.*, vol. 4, no. 2, pp. 490–530, 2005.

[21] G. R. Cross and A. K. Jain, "Markov random field texture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 1, pp. 25–39, 1983.

[22] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *ICCV*, vol. 2. IEEE, 1999, pp. 1033–1038.

[23] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *CVPR*, vol. 2. IEEE, 2005, pp. 60–65.

[24] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000, pp. 479–488.

[25] M. Ashikhmin, "Synthesizing natural textures," in *Interactive 3D graphics*. ACM, 2001, pp. 217–226.

[26] R. Bornard, E. Lecan, L. Laborelli, and J.-H. Chenot, "Missing data correction in still images and image sequences," in *Proceedings of the tenth ACM international conference on Multimedia*, 2002, pp. 355–361.

[27] I. Drori, D. Cohen-Or, and H. Yeshurun, "Fragment-based image completion," in *ACM T. Graphic.*, vol. 22, no. 3. ACM, 2003, pp. 303–312.

[28] P. Pérez, M. Gangnet, and A. Blake, "Patchworks: Example-based region tiling for image editing," Microsoft Research, MSR-TR-2004-04, Tech. Rep., 2004.

[29] O. Le Meur, J. Gautier, and C. Guillemot, "Examplar-based inpainting based on local geometry," in *ICIP*, Brussel, Belgium, 2011, pp. 3401–3404. [Online]. Available: http://hal.inria.fr/inria-00628074

[30] Z. Xu and J. Sun, "Image inpainting by patch propagation using patch sparsity," *IEEE T. Image Process.*, vol. 19, no. 5, pp. 1153–1165, 2010.

[31] R. Martinez-Noriega, A. Roumy, and G. Blanchard, "Exemplar-based image inpainting: Fast priority and coherent nearest neighbor search," in *MLSP*. IEEE, 2012, pp. 1–6.

[32] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher, "Simultaneous structure and texture image inpainting," *IEEE T. Image Process.*, vol. 12, no. 8, pp. 882–889, 2003.

[33] J.-L. Starck, M. Elad, and D. L. Donoho, "Image decomposition via the combination of sparse representations and a variational approach," *IEEE T. Image Process.*, vol. 14, no. 10, pp. 1570–1582, 2005.

[34] J. Jia and C.-K. Tang, "Inference of segmented color and texture description by tensor voting," *IEEE Trans. Pattern Anal.*, vol. 26, no. 6, pp. 771–786, 2004.

[35] F. Cao, Y. Gousseau, S. Masnou, and P. Pérez, "Geometrically guided exemplar-based inpainting," *SIAM J. Imag. Sciences*, vol. 4, no. 4, pp. 1143–1179, 2011.

[36] Y. Wexler, E. Shechtman, and M. Irani, "Space-time completion of video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 3, pp. 463–476, Mar. 2007.

[37] N. Komodakis and G. Tziritas, "Image completion using efficient belief propagation via priority scheduling and dynamic pruning," *Image Processing, IEEE Transactions on*, vol. 16, no. 11, pp. 2649–2661, 2007.

[38] Y. Pritch, E. Kav-Venaki, and S. Peleg, "Shift-map image editing," in *International Conference on Computer Vision*. IEEE, 2009, pp. 151–158.

[39] A. Mansfield, M. Prasad, C. Rother, T. Sharp, P. Kohli, and L. J. Van Gool, "Transforming image completion." in *BMVC*, 2011, pp. 1–11.

[40] P. Arias, V. Caselles, and G. Facciolo, "Analysis of a variational framework for exemplar-based image inpainting," *Multiscale Modeling & Simulation*, vol. 10, no. 2, pp. 473–514, 2012.

[41] K. He and J. Sun, "Statistics of patch offsets for image completion," in *European Conference on Computer Vision*. Springer-Verlag, 2012, pp. 16–29.

[42] A. Newson, M. Fradet, P. Pérez, A. Almansa, and Y. Gousseau, "Towards fast, generic video inpainting," in *European Conference on Visual Media Production, 2013. CVMP 2013.*, 2013, pp. 1–8.

[43] O. Le Meur, M. Ebdelli, and C. Guillemot, "Hierarchical super-resolution-based inpainting," *IEEE T. Image Process.*, vol. 22, no. 10, pp. 3779–3790, 2013.

[44] S. Di Zenzo, "A note on the gradient of a multi-image," *Comput. Vision Graph.*, vol. 33, no. 1, pp. 116–125, 1986.

[45] J. Sun, L. Yuan, J. Jia, and H.-Y. Shum, "Image completion with structure propagation," *ACM T. Graphic.*, vol. 24, no. 3, pp. 861–868, Jul. 2005.

[46] A. Wong and J. Orchard, "A nonlocal-means approach to exemplar-based inpainting," in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*. IEEE, 2008, pp. 2600–2603.

[47] C. Guillemot, M. Turkan, O. Le Meur, M. Ebdelli *et al.*, "Object removal and loss concealment using neighbor embedding," *Eurasip Journal on Signal Processing: Image Communication*, 2013.

[48] N. Kawai, T. Sato, and N. Yokoya, "Image inpainting considering brightness change and spatial locality of textures and its evaluation," in *Proc. of the 3rd PSIVT*, ser. PSIVT '09, 2009, pp. 271–282.

[49] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D*, vol. 60, no. 1, pp. 259–268, 1992.

[50] Y. Meyer, *Oscillating patterns in image processing and nonlinear evolution equations: the fifteenth Dean Jacqueline B. Lewis memorial lectures*. American Mathematical Soc., 2001, vol. 22.

[51] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "PatchMatch: A randomized correspondence algorithm for structural image editing," *ACM T. Graphic.*, vol. 28, no. 3, Aug. 2009.

[52] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen, "Interactive digital photomontage," in *ACM Transactions on Graphics*, vol. 23, no. 3. ACM, 2004, pp. 294–302.

[53] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM T. Graphic.*, vol. 22, no. 3, pp. 313–318, Jul. 2003.

[54] K. He and J. Sun, "Computing nearest-neighbor fields via propagation-assisted kd-trees," in *CVPR*. IEEE, 2012, pp. 111–118.

(a) Exemplar-based inpainting result. (b) Map of local artifacts strength. (c) Exemplar-based inpainting + our spatial patch blending algorithm.

Fig. 11. Artifact detection pipeline from the inpainting result (left) to the spatially blended image result (right).

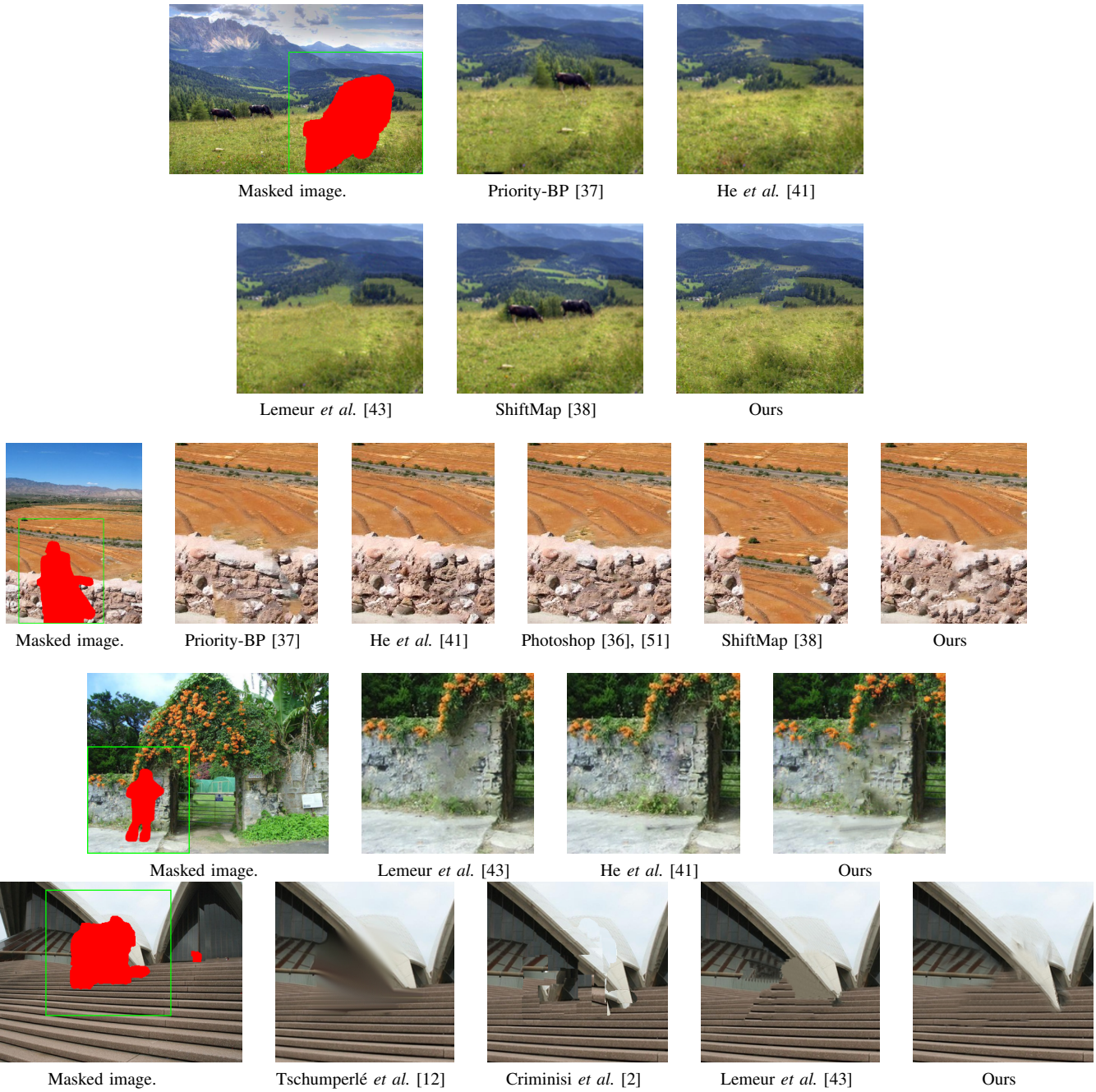


Fig. 14. Comparisons of inpainted results with several state-of-the-art methods on difficult natural images, namely *cow*, *wall*, *yard*, and *Sydney opera*.