A Fast Spatial Patch Blending Algorithm for Artefact Reduction in Pattern-based Image Inpainting

Maxime Daisy, David Tschumperlé, Olivier Lézoray *
GREYC Laboratory (CNRS UMR 6072), Image Team, 6 Bd Maréchal Juin, 14050 Caen/France

Figure 1: *Illustration of our proposed spatial patch blending algorithm for image inpainting. From left to right : color image with area to reconstruct, reconstruction result with the inpainting algorithm from [Criminisi et al. 2004], our reconstruction result.*

Abstract

We propose a fast and generic spatial patch blending technique that can be embedded within any kind of pattern-based inpainting algorithm. This extends the works of [Daisy et al. 2013] on the visual enhancement of inpainting results. We optimize this blending algorithm so that the processing time is roughly divided by a factor ten, without any loss of perceived quality. Moreover, we provide a free and simple-to-use software to make this easily reproducible.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation— [I.3.4]: Computer Graphics—Graphics Utilities I.4.4 [Image Processing and Computer Vision]: Restoration— [I.4.9]: Image Processing and Computer Vision—Applications

Keywords: inpainting, spatial, patch, blending, patch-based

1 Introduction

Filling unkown or removing undesired contents from images, known as image inpainting, is a widely used tool today. Movie producers for example, use it to remove microphones or scratches from new and older movie sequences. As this kind of reconstruction tool is employed by users that want their images to look more realistic, it must obviously not damage the perceptual and visual quality of the processed images. In the state of the art, there mainly exist two kinds of inpainting methods. Geometry-based methods [Masnou and Morel 1998; Bertalmio et al. 2000; Tschumperlé and Deriche 2005] provide techniques to propagate image structures by extrapolating the local geometry. Unfortunately, these methods are often unable to synthesize non-local structures like textures. On the contrary, in pattern-based methods [Criminisi et al. 2004; Le Meur et al. 2011], user-selected image areas are reconstructed by copying patches from the known image zones, to those unkown. These methods work well to reconstruct textures. Even with some variations, like avering several patches [Le Meur et al. 2011], they generally do not provide good result in terms of global geometry consistency. Hybrid methods also exist [Sun et al. 2005], but there are always some cases where reconstruction let some artefacts appear. Recently, [Daisy et al. 2013] introduced a spatial patch blending technique to perceptually reduce these reconstruction artefacts, without any changes in the way geometry is inpainted. Unfortunately, this method cannot be used for production work due to computational burden and memory overload.

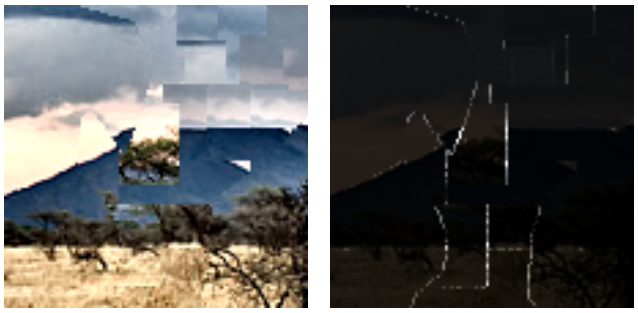
The paper addresses these two issues and is organized as follows. First, the principle of spatial patch blending is presented through a complete summary of the method. Then, we redesign the blending algorithm and show the various improvements it implies. Finally, we illustrate the relevance of our approach by commented results and comparisons with some state-of-the-art methods.

2 Patch Blending Context and Previous Work

In [Daisy et al. 2013] was proposed a method that allows reducing artefacts produced by any patch-based inpainting algorithm [Criminisi et al. 2004; Le Meur et al. 2011] in an image I. A result image J is produced where possible inter-patch seams and inconsistencies are cleverly hidden, rendering the image perceptually more pleasant. In this method, it is proposed to modify the process of any patch-based inpainting algorithm so that it provides additional information. The latter is used to perform the two steps of our artefact reduction technique, namely: 1) the artefact detection 2) the spatial patch blending.

• **Artefact Detection**. This first step of this method is empirically based on the two following hypothesis. For a reconstructed image I which reconstruction patch locations are stored in a map $\mathcal{U} : p \in I \mapsto q \in I$, it is hypothesized that: 1) there are sharp color or luminosity variations where artefacts are located, and 2) patches from remote locations seem probably differents. The idea is first to combine the latter to estimate a set of points $\mathcal{E}$ where the strongest artefacts are located. Then, the map $\sigma : p \in I \mapsto \sigma(p) \in \mathbb{R}$ of blending amplitudes is computed, and gives to each point p inside a mask M a weight depending on its distance to the nearest artefact locations and strengths (cf. Fig. 2, and (3) of [Daisy et al. 2013]).

*e-mails:
{Maxime.Daisy, David.Tschumperle, Olivier.Lezoray}@ensicaen.fr



(a) A reconstructed image where artefacts are to be detected.

(b) The superimposed set $\mathcal{E}$ with the associated amplitudes σ result of the artefact detection in 2(a).

Figure 2: *Illustration of the artefact detection result.*

Figure 3: *Results of a spatial patch blending (zoomed). From left to right : masked image, result of patch-based inpainting [Criminisi et al. 2004], result of [Daisy et al. 2013].*

• **Spatial Patch Blending**. In classic patch-based inpainting methods, the reconstruction of an image is a kind of patchwork. Patches are iteratively extracted from the image, cut up, and the remaining pieces are pasted inside M to complete the given image. The main idea of the spatial patch blending is to point out the fact that parts of the individual patches are discarded during sequential compositing, but these parts contain valuable information that could have been used if a different insertion order had been used. In this method, the scrapped offcuts are kept and spatially blended in order to reduce seams between the pieces of patches pasted side by side. This method is defined as a pixelwise process, and for each point $p \in M$, the set Ψ_p of patches overlapping at p is extracted. Then, a combination of all the pixels where all the patches $\psi_q \in \Psi_p$ overlap is computed as follows:

$$J(p) = \frac{\sum\limits_{\psi_q \in \Psi_p} w(q,p)\ \psi_q(p-q)}{\varepsilon + \sum\limits_{\psi_q \in \Psi_p} w(q,p)} \qquad (1)$$

The gaussian weight function $w(p,q) = e^{\frac{d(q,p)^2}{\sigma^2}}$ defines the way patches are blended together during the process. This function strongly depends on the distance function d. In [Daisy et al. 2013], they have used the minimal distance from the point p to every point in the piece of pasted patch ψ_q. As shown in Fig. 3, this method provides clearly good results in terms of artefact reduction regarding a classical patch-based inpainting result. On the other hand, the memory usage for storing the map $\mathcal{U}$ is too important. Then, even if it is reasonable as compared to these of the inpainting process, the computation time does not allow this method to be used easily and interactively. The main reason is that as many distance maps as reconstruction patches have to be computed. This makes the computation time to be very dependant of the mask size used for the inpainting. In addition, the size of $\mathcal{E}$ is a little bit overestimated by the artefact detection. In this pixel per pixel process, this causes the computation to increase noticeably. The weaknesses of [Daisy et al. 2013] have brought us to redesign some parts of their algorithm to make it faster while maintaining the good perceptual quality of the results. The contributions we propose through this paper are mainly based on the enhancement of the spatial patch blending algorithm in terms of time consumption, but also memory usage.

3 Enhanced Spatial Patch Blending

We propose here a spatial patch blending algorithm for pattern-based inpainting algortihms. Firstly, this method is described as it is, and then we discuss the different enhancements in comparison with the method of [Daisy et al. 2013].

• **Spatial blending reformulation**. At first sight, the method we propose seems to be very different from [Daisy et al. 2013]. Rather than independently computing each final pixel $J(p)$ by looking for every $p \in M$, the set of local features that allows computing (1), we propagate each patch feature at once on the pasting neighbourhoods of p. Loop on each point p is replaced by a loop on all patches ψ_q pasted in I during the inpainting. This second loop needs much less computing iterations (approximatly $n^2/2$ less where $n \times n$ is the inpainting patch dimension). This loop factorization is theorically possible only if the bandwidth $\sigma(p)$ of the blending is considered as constant on the whole image. This obviously not the case. To do so, a multiscale approach is adopted. The loop on patches is repeated as many times as the number of different scales that can be considered in the values of σ (we have quantized these values into N scales). As N can be chosen small enough (typically of about ten scales, smaller values would leads to some discontinuities in the blending), the looping repetition factor due to the multi-scale aspect of our algorithm remains much less important than the average gain of $n^2/2$ at a specified scale. This makes the final algorithm very interesting in terms of complexity compared to the approach of [Daisy et al. 2013]. Algorithm 1 details the whole principle of our multi-scale spatial blending method.

One can notice that this method of blending acts like a *post-processing* of the image inpainting result, but requires to modify the considered patch-based inpainting algorithm, for the reconstruction patches locations and the reconstruction points to be stored.

• **Differences with the previous approach**. The differences of our method compared to the approach of [Daisy et al. 2013] are the following:

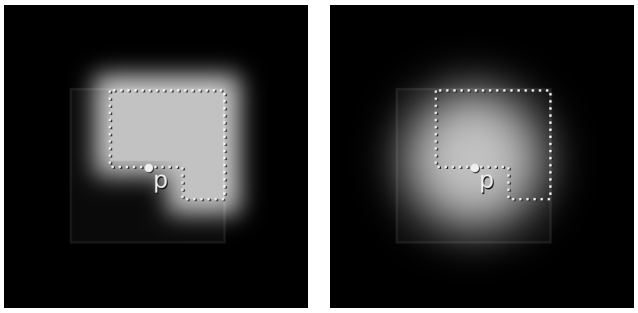
Quantized spatial blending scales: Our optimized algorithm considers a quantized version of the spatial blending amplitude map σ. The set of the blending results J_s are computed for each scale $\sigma_s \in [1, N] \subset \mathbb{N}$ and are then merged in a final image J. This image contains pixels of $J_1, J_2, \ldots, J_N$ depending on the local (quantized) scale defined in $\tilde{\sigma}(p)$. The storage of all blending scales J_s can be easily avoided by transferring directly all the pixels computed at a scale s to the final image J. In this case, the last loop of Algorithm 1 has to be done in the main scale loop (line 5).

Modified weight function: The spatial patch blending is locally performed as a linear combination of all the patches that would have overlapped with a different inpainting order. One can demonstrate that with this new algorithm, the weighting function $w(p,q)$ of each patch (also used in (1)) depends only on the distance from a point to the neighbour reconstruction points rather than the distance from a point to a piece of pasted patch (as described in [Daisy et al. 2013]).

Algorithm 1: Fast spatial patch blending for inpainting algorithms.

Input: Inpainted image I, Inpainting mask M, Number of scales N.
Output: Image J with spatially blended patches.

Initialize P = Ordered list of original patch center locations (p, q) pasted in M during the inpainting;
Initialize C = Ordered list of patch pasting locations (x, y) of during inpainting;
Initialize σ = Estimated local blending amplitude (section 2);
Initialize $\tilde{\sigma}$ = Uniform quantization of σ in N levels $(\sigma_1, \ldots, \sigma_N)$;
`// Computation of the spatial blending levels` J_s
`// for differents` σ_s
for $s \in [1, N] \subset \mathbb{N}$, **do**
    Initialize J_s = Result color image of the blending at scale s, initialized to 0 for all pixel in M, and $I(p)$ elsewhere;
    Initialize A = Scalar accumulation image of the size of I, initialized to 0 for all pixels in M, and 1 elsewhere;
    Initialize ϕ = Image of size $m \times m$, containing a centered Gaussian of variance σ_s;
    for $k \in P$ **do**
        Add the patch of size $m \times m$ of I located at $P(k)$ to the image J_s at $C_{(k)}$;
        Add the image ϕ of the gaussian weights in A at $C_{(k)}$;
    Divide J_s by A (normalisation of the added colors).
`// Combine all the blending scales`
`// in a result image.`
for $p \in M$, **do**
    $s = \tilde{\sigma}_{(p)}$;
    $J_{(p)} = J_{s(p)}$;



(a) Weights used in [Daisy et al. 2013].

(b) Weights use in our new method.

Figure 4: *Illustration of the difference between weights of [Daisy et al. 2013] (a), and those used in our fast spatial patch blending algorithm (b).*

This is mainly thanks to this approximation that our optimized spatial blending version of the algorithm of [Daisy et al. 2013] can be reformulated. From an experimental point of view, one can notice that the difference between the results produced with the two weight functions are very difficult to see in the final blending results.

Mask-external spatial patch blending: In Algorithm 1, the spatial blending is naturally extended to the outside of the inpainting mask M. In terms of visual appeal, this is very interesting since a smooth transition is created between the known colors and these of the reconstructed area. All the results presented in the following section take advantage of this special feature. To respect the classic inpainting formalism, one can constraint pixels from outside the mask not to be modified by our spatial patch blending (by copying all known pixels from I to the final image J at the end of the process).

Performance improvement: The gain performance of our approach as compared to [Daisy et al. 2013], and the comparison with the state-of-the-art approaches of [Criminisi et al. 2004] (inpainting without spatial patch blending) and Photoshop (very fast, based on [Wexler et al. 2007; Barnes et al. 2009]) is illustrated Fig. 5. In order to show the efficiency of our new method, we have made some experimentations. Fig. 5 summarizes the results on a set of medium-sized image [1], and mainly gives us three interesting informations. First, the gain of time between the approach of [Daisy et al. 2013] and our method depends on the kind of processed images (mainly depending on the size of M), but is very significant in each case (from 6 to 30 times faster for the presented examples). Then, there is no meaningful difference of computation time between method in [Criminisi et al. 2004] and ours. This means that there is no additional cost to process our spatial patch blending algorithm after a classic patch-based inpainting. Also, one can see that the content-aware filling algorithm [Wexler et al. 2007; Barnes et al. 2009] provided in Photoshop is noticeably faster than our method, but is most likely using material accelerations like GPU processing or multi-core programming. This is not the case of our method, provided with standard C++ implementation with no acceleration.

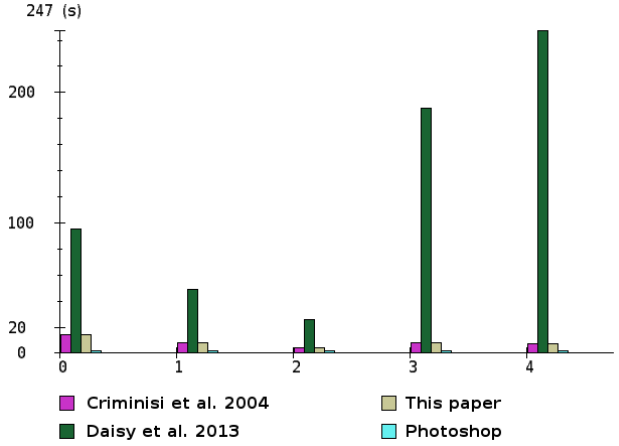


Figure 5: *Illustration of execution time comparison between our method, method of [Daisy et al. 2013], with state-of-the-art methods. The less, the better.*

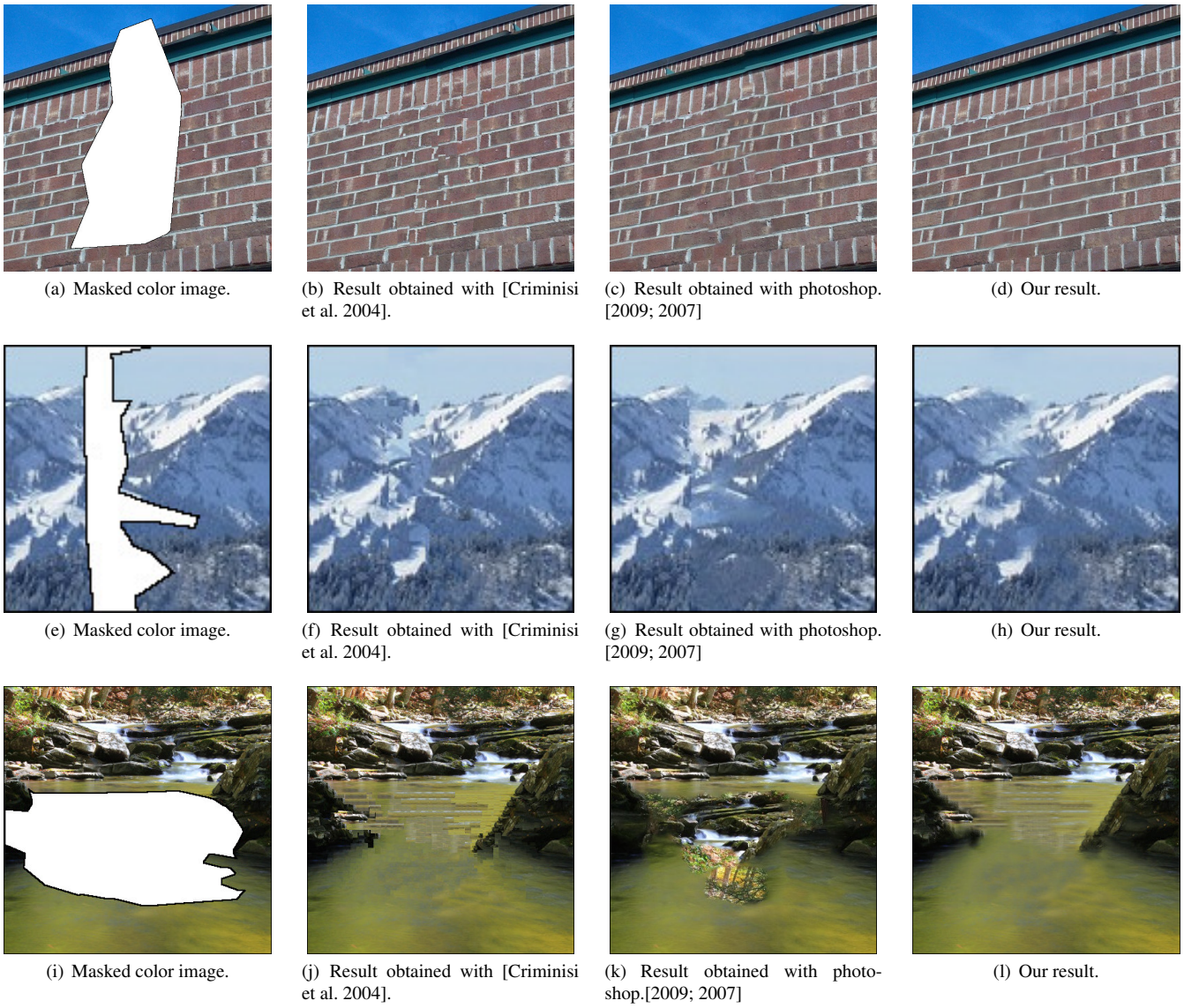
4 Results and Reproducibility

Some results provided by our method are illustrated Fig. 6 and compared to state-of-the-art methods. Spatial patch blending is clearly demonstrated through our examples and our way of making it faster allows now this method to be used interactively. In addition, a software integration of our method has been made and the source code is now available to the community, making our fast spatial patch blending algorithm fully reproducible:

• The source code of our technique is available as a function named *inpaint_patch()* in G'MIC [Tschumperlé 2013] source codes.

• A dedicated filter has been added to the G'MIC plugin for the open source GIMP[2] software, allowing non specialist people to use it easily thanks to an enhanced graphical user interface.

[1] http://daisy.users.greyc.fr/@publications:id=fspba.dtl.2013
[2] http://www.gimp.org/



(a) Masked color image. (b) Result obtained with [Criminisi et al. 2004]. (c) Result obtained with photoshop. [2009; 2007] (d) Our result.

(e) Masked color image. (f) Result obtained with [Criminisi et al. 2004]. (g) Result obtained with photoshop. [2009; 2007] (h) Our result.

(i) Masked color image. (j) Result obtained with [Criminisi et al. 2004]. (k) Result obtained with photoshop.[2009; 2007] (l) Our result.

Figure 6: *Comparison with several state-of-the-art methods (zoomed).*

References

BARNES, C., SHECHTMAN, E., FINKELSTEIN, A., AND GOLDMAN, D. B. 2009. Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM Trans. Graph. 28*, 3 (July), 24:1–24:11.

BERTALMIO, M., SAPIRO, G., CASELLES, V., AND BALLESTER, C. 2000. Image inpainting. In *Proc. of the 27th annual SIGGRAPH conference*, SIGGRAPH '00, 417–424.

CRIMINISI, A., PÉREZ, P., AND TOYAMA, K. 2004. Region filling and object removal by exemplar-based image inpainting. *IEEE Trans. Im. Proc. 13*, 9 (Sept.), 1200–1212.

DAISY, M., TSCHUMPERLÉ, D., AND LÉZORAY, O. 2013. Spatial patch blending for artefact reduction in pattern-based inpainting techniques. In *Int. Conf. on Computer Analysis of Images and Patterns(CAIP)*, vol. LNCS 8048, 523–530.

LE MEUR, O., GAUTIER, J., AND GUILLEMOT, C. 2011. Examplar-based inpainting based on local geometry. In *ICIP*, 3401–3404.

MASNOU, S., AND MOREL, J.-M. 1998. Level lines based disocclusion. In *ICIP (3)*, 259–263.

SUN, J., YUAN, L., JIA, J., AND SHUM, H.-Y. 2005. Image completion with structure propagation. *ACM Trans. Graph. 24*, 3 (July), 861–868.

TSCHUMPERLÉ, D., AND DERICHE, R. 2005. Vector-valued image regularization with pdes: A common framework for different applications. *IEEE Trans. PAMI 27*, 4, 506–517.

TSCHUMPERLÉ, D. 2013. G'MIC : Greyc's magic for image computing. *http://gmic.sourceforge.net/*.

WEXLER, Y., SHECHTMAN, E., AND IRANI, M. 2007. Space-time completion of video. *IEEE Trans. Pattern Anal. Mach. Intell. 29*, 3 (Mar.), 463–476.