Un algorithme rapide de fondu spatial pour la réduction d'artefacts visuels des méthodes d'inpainting "basés patch"

M. Daisy, D. Tschumperlé et O. Lezoray

Laboratoire GREYC (CNRS UMR 6072), Equipe IMAGE, 6 Bd Maréchal Juin, 14050 Caen, France.
{Maxime.Daisy, David.Tschumperle, Olivier.Lezoray}@ensicaen.fr

Résumé

Nous proposons une technique rapide et générique de fondu spatial de patch, intégrable dans tout type d'algorithme d'"inpainting" basé motif (reconstruction de régions manquantes dans des images, par copier/coller de patchs d'images). Cette contribution, qui est la suite logique de nos précédents travaux sur l'amélioration visuelle des méthodes d'"inpainting" [1], optimise le temps de rendu (d'un facteur dix en moyenne) sans dégradation de qualité sur les résultats obtenus. L'algorithme que nous décrivons ici est simple à mettre en oeuvre et suit une démarche de recherche reproductible, puisque son code source est accessible sous licence libre, et une interface logicielle (libre également) a été développée pour le rendre utilisable par tout un chacun.

Mots clefs

Reconstruction d'images basée motifs, Inpainting par patchs, Fondu spatial de motifs, Recherche reproductible.

1 Introduction

La reconstruction automatique de zones inconnues ou non désirées dans des images, aussi appellée "inpainting" d'image, est depuis quelques années un sujet de recherche en plein essor avec de nombreuses applications pratiques. On peut l'utiliser pour retirer des éléments génants présents dans une image (tels qu'un microphone ou un câble par exemple, dans le domaine de la production cinématographique), ou reconstruire des zones entières d'images détériorées (dues à des rayures ou des dégradations chimiques sur des photographies ou des vieux films). Comme ce genre d'algorithme est utilisé *in fine* pour améliorer la qualité visuelle des images, on cherche à tout prix à rendre invisible leurs actions, en minimisant l'introduction d'*artefacts visuels de reconstruction*, qui peuvent parfois nécessiter des post-traitements manuels fastidieux.

Dans l'état de l'art, il existe principalement deux philosophies distinctes de reconstruction : les méthodes de complétion géométrique [2, 3] qui essayent d'extrapoler les structures de l'image en suivant une géométrie de reconstruction définie à un niveau *pixelique*, et les méthodes d'inpainting "basée motifs" [4, 5], qui consistent à recoller itérativement de petites portions d'images (*patchs*) dans les zones à reconstruire. Ces dernières techniques font aujourd'hui l'unanimité quant à leurs performances pour traiter le problème difficile de la reconstruction automatique de larges zones texturées dans les images.

Figure 1 – *Minimisation des artefacts visuels de reconstruction d'inpainting basé motifs. De haut en bas : Image avec zone à reconstituer, Image "inpaintée" avec [4], Image "inpaintée" avec l'algorithme de fondu spatial de patch proposé dans ce papier.*

Cependant, elles ne parviennent pas toujours à agencer des patchs de manière à ce qu'ils se recollent parfaitement, et cela peut introduire des artefacts visuels importants à l'endroit des recollements, notamment sur le squelette des masques définissant les zones à reconstruire (Fig.1). Des approches hybrides géométrie/motifs existent [6, 7, 8] mais elles ne résolvent pas de manière définitive l'apparition de ces portions de reconstruction "pathologiques".

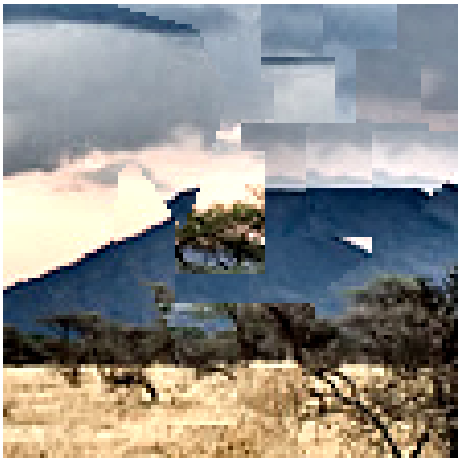
Dans ce papier, nous proposons une méthode rapide et simple à mettre en oeuvre de *fondu spatial de patchs* afin de minimiser visuellement ce type d'artefacts de reconstruction, que l'on rencontre malheureusement dans tous types d'algorithmes d'inpainting basé motifs. Plus précisemment, nous proposons une version simplifiée et optimisée d'une méthode de fondu, basée sur nos travaux précédents [1], et qui a l'avantage d'introduire un gain de temps de calcul important (d'un facteur 10 en moyenne). Dans un premier temps, ces travaux sont brièvement rappelés (section 2), puis nous introduisons notre nouvel algorithme de fondu spatial de motif (section 3), et illustrons ses performances visuelles et computationnelles en le comparant avec quelques algorithmes de l'état de l'art.

2 Fondu spatial de patch

Dans [1], nous avons proposé une méthode originale permettant de réduire les artefacts visuels dans les algorithmes d'inpainting d'image basés motifs tels que [4, 5]. Ces arterfacts surviennent principalement lors de recollements côte à côte de patchs qui, malheureusement, proviennent de localisation très différentes de l'image et qui introduisent donc des discontinuités assez nettes d'intensité, de couleur ou de texture dans la zone à reconstruire. Notre méthode ne cherche pas à modifier ces algorithmes d'inpainting pour que ces artefacts visuels n'apparaissent plus - ce problème étant particulièrement difficile à résoudre - mais au contraire cherche à minimiser *après coup* l'impact visuel de ces mauvaises reconstructions locales. Elle se base sur deux étapes distinctes, à savoir : (1) la détection des artefacts visuels, (2) l'application d'un fondu spatial de patch adapté localement à l'intensité des artefacts détectés.

2.1 Détection des artefacts visuels

Un artefact visuel définit une zone ou les patchs se sont localement mal recollés, et un point $p = (x, y)$ de l'image $I : \Omega \rightarrow \mathbb{R}^n$ est donc labelisé comme "mal reconstruit" (point de rupture) s'il vérifie *simultanément* ces deux conditions : (1) $\nabla I_{(p)}$ est élevé, ce qui signifie que le point p est placé sur une discontinuité des structures de l'image. (2) Localement, les patchs recollés autour de p proviennent d'endroits très différent, ce qui peut se détecter par la mesure de la divergence de la carte des positions originales des patchs recollés $\mathcal{U}_{(p)}$. Ainsi, si un point p vérifie (1) et (2), il se situe nécéssairement à la frontière entre deux patchs recollés et ce recollement a créé une discontinuité suffisamment élevée pour qu'elle puisse entrainer un artefact visuel. Ces deux contraintes définissent un ensemble de points d'artefacts visuels $\mathcal{E}$ détectés. Par la suite, on définit la carte $\sigma : \Omega \rightarrow \mathbb{R}$ des amplitudes locales de fondu spatial de patch, calculées comme décroissantes exponentiellement en fonction de la distance aux points de rupture $\mathcal{E}$ les plus proches (Fig.2.1).



(a) Images "inpaintée" où les points d'artefacts doivent être détectés.

(b) Ensemble des points $\mathcal{E}$ détectés. L'intensité lumineuse des points est proportionnelle à l'amplitude de la rupture estimée.

Figure 2 – *Illustration de notre technique de détection des artefacts visuels de reconstruction.*

2.2 Fondu spatial de patchs

L'utilisation d'un algorithme de fondu spatial de patch dans une méthode d'inpainting basée motifs peut se justifier de la manière suivante : si deux patchs côte à côte ont été mal positionnés et génèrent un artefact visuel à l'endroit du recollement, il est avantageux de les faire "fondre" l'un dans l'autre de manière spatialement progressive afin de minimiser l'impression locale de discontinuité. De la même manière qu'un fondu d'images temporel permet de générer des transitions plus douces entre différentes frames d'une séquence vidéo, notre fondu spatial de patch va être capable de limiter l'impact visuel de la discontinuité des

structures locales recollées, sans dégrader la structure du contenu interne des patchs recollés (texture préservée notamment). C'est très différent de ce que peut générer un algorithme de lissage effectué par moyennage de motifs (comme proposé dans [5] pour l'inpainting), puisqu'ici on va être capable de générer des moyennages de points de différents patchs à *une échelle pixellique, locale au patch*, en prenant en compte la configuration spatiale à la fois de l'artefact, mais aussi du recollement. Cela ne peut se réaliser qu'une fois que la première phase d'inpainting a été complétée. Ce mélange spatial de patch se réalise en chaque point $p \in \Omega$ par la combinaison linéaire suivante :

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

où $J_{(p)}$ est la couleur du point résultat, Ψ_p est l'ensemble des patch ψ_q recollés autour de p, $w(q,p) = e^{-\frac{d(q,p)^2}{\sigma^2}}$ sont des poids gaussiens qui régulent le fondu spatial en prenant en compte la distance spatiale minimale $d(q,p)$ entre le point p et les différentes points q des patch ψ_q de I recollés dans le voisinage de p (détails dans [1]). Comme on peut le voir sur les Fig.1 et 3, le gain visuel qui découle de ce mélange spatial de patch est particulièrement intéressant, alors même que la procédure de recollement des patchs utilisée par les deux algorithmes est strictement la même.

Figure 3 – *Illustration du fondu spatial de patchs pour l'inpainting (zoom d'une image plus grande). De gauche à droite : Image avec zone à reconstituer, Image "inpaintée" avec [4], Image "inpaintée" avec notre mélage spatial de motifs [1].*

Néanmoins, la méthode proposée dans [1] n'est pas satisfaisante en ce qui concerne les points suivants.

• **Rapidité d'exécution :** Pour chaque point $p \in \Omega$ de l'image I, on doit reconstruire la liste Ψ_p des patchs uniques ψ_q recollés autour de p, puis moyenner les couleurs $\psi_q(p-q)$ de chacun de ces patchs qui auraient pu se trouver en p. Construire Ψ_p nécessite le parcours d'un voisinage carré de p, et a donc la complexité d'une convolution de I par un noyau discret.

• **Occupation mémoire :** L'algorithme d'inpainting par patch utilisé doit être modifié pour se rappeller en chaque point reconstruit p la position du patch d'origine $\mathcal{U}_{(p)}$ qui a été recollé, mais aussi l'offset (i,j) du point recollé à l'intérieur même du patch recollé. On doit donc stoker deux cartes de positions de la taille de l'image d'origine.

Ces deux défauts peuvent devenir rédhibitoires si l'on a un grand nombre d'images à traiter (séquences vidéos stéreoscopiques par exemple), et nous avons donc développé un algorithme rapide de fondu spatial de patch, basé sur des approximations de l'éq.1 qui permettent une accélération du temps de traitement par un facteur 10 en moyenne, tout en maintenant une qualité visuelle de reconstruction comparable à notre méthode précédente. Cette optimisation nous a permis de développer un algorithme suffisamment rapide pour être utilisé en production, avec une implémentation standard en C++ (sans accélération GPU ou parallélisme). C'est cet algorithme que nous décrivons dans la section suivante.

3 Algorithme rapide de fondu

Nous proposons ici un nouvel algorithme de fondu spatial de patchs pour l'inpainting basé motifs. Cet algorithme est d'abord décrit tel quel, puis nous discutons des différences qu'il apporte vis-à-vis de la méthode décrite en section précédente.

3.1 Reformulation du fondu spatial

De prime abord, la méthode que nous proposons peut paraître totalement différente du processus précédent : Plutôt que d'essayer de calculer chaque pixel résultat $J_{(p)}$ de manière indépendante, en recherchant pour chaque point de l'image $p \in \Omega$, l'ensemble des caractéristiques locales permettant de calculer l'Eq.(1), nous allons propager les caractéristiques de chaque patch en une fois, sur les voisinages de recollement. Par conséquent, nous transformons notre boucle sur chaque point $p \in \Omega$ en une boucle sur chaque patch ψ_q recollé dans I pendant le processus d'inpainting proprement dit. Cette deuxième boucle nécéssite beaucoup moins d'itérations de calcul (approximativement $n^2/2$ moins d'itérations où $n \times n$ est la dimensions des patchs considérés pour l'inpainting). Cette factorisation de boucle ne serait théoriquement possible que si la variance $\sigma_{(p)}$ du fondu était considérée constante sur toute l'image I, ce qui n'est évidemment pas le cas.
On adopte donc une approche *multi-échelle* en répétant cette boucle de patchs autant de fois que l'on considère d'échelles différentes dans les valeurs d'amplitudes de fondu σ (quantifié ainsi en N valeurs d'échelles différentes). Comme N peut être suffisamment petit (typiquement une dizaine d'échelles quantifiées σ suffisent), le facteur de répétition de boucle dû à l'aspect multi-échelle de l'algorithme reste malgré tout très en deça du gain moyen de $n^2/2$ gagné sur la complexité de chaque boucle à une

échelle donnée, ce qui rend l'algorithme final très intéressant en termes de complexité par rapport à l'approche précédente de la section 2.
L'algorithme 1 détaille sous forme de pseudo-code le principe de fondu spatial multi-échelle que nous proposons :

Algorithme 1 : Fondu spatial rapide pour l'inpainting

Input : Image inpaintée I, Masque d'inpainting M, Nombre d'échelles N.
Output : Image J avec patchs mélangés spatialement.

Initialise P = Liste ordonnée des positions (p, q) des centres des patchs d'origines de I recollés dans M lors de l'inpainting;
Initialise C = Liste ordonnée des positions (x, y) de recollage des patchs lors de l'inpainting;
Initialise σ = Carte d'amplitude locale de fondu estimé (voir section 2.1);
Initialise $\tilde{\sigma}$ = Quantification uniforme de σ en N niveaux $(\sigma_1, ..., \sigma_N)$;
`// Calcul du rendu des fondu spatiaux` J_s `pour plusieurs échelles différentes` σ_s`.`
for $s \in [1, N] \subset \mathbb{N}$, **do**
    Initialise J_s = Image couleur résultat du fondu pour l'échelle s, initialisée à 0 pour les pixels de M, et $I(p)$ ailleurs;
    Initialise A = Image scalaire d'accumulation, de même taille que I, initialisée à 0 pour les pixels de M, et 1 ailleurs;
    Initialise ϕ = Image de taille $M \times M$, contenant une gaussienne centrée de variance σ_s;
    for $k \in P$ **do**
        Additionne le patch de taille $M \times M$ de I localisé en $P(k)$, à l'image J_s à la position $C_{(k)}$;
        Additionne l'image ϕ des poids gaussiens dans l'image A à la position $C_{(k)}$;
    Divise J_s par A (normalisation des couleurs additionnées).
`// Fusionne l'ensemble des échelles de fondus calculés en une image résultat.`
for $p \in \Omega$, **do**
    $s = \tilde{\sigma}_{(p)}$;
    $J_{(p)} = J_{s(p)}$;

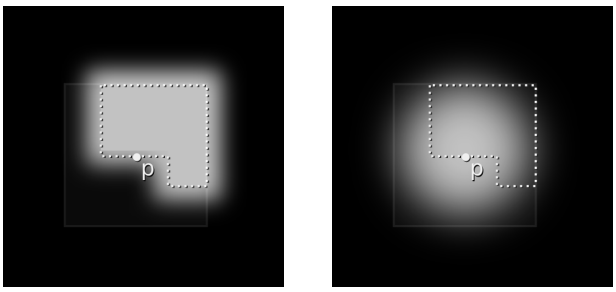
Notons que cette méthode de fondu agit comme *post-traitement* du résultat de l'image inpaintée, mais qu'elle *nécessite de modifier* l'algorithme d'inpainting original pour stocker les coordonnées des patchs qui ont été recollés, ainsi que leurs positions de recollement.

3.2 Différences avec l'approche précédente

Par rapport à l'approche précédente (Eq.(1)), on peut montrer que les différences se résument à :

• **Echelles quantifiées de fondu spatial :** Notre algorithme optimisé considère une version quantifiée de la carte d'amplitudes σ de fondu spatial de patch. On calcule l'ensemble des résultats de fondu J_s pour chaque échelle $\sigma_s \in [1, N]$, puis on fusionne les résultats sous forme d'une image finale J contenant les pixels de $J_1, J_2, ..., J_N$ suivant l'échelle locale (quantifiée) désirée définie dans $\tilde{\sigma}_{(p)}$. On peut avantageusement éviter le stockage en mémoire de toutes ces échelles calculées J_s, en transférant les pixels concernés directement dans l'image finale J juste avant de calculer une nouvelle échelle. La dernière boucle de l'Algorithme 1 se retrouve alors dans la boucle principale du parcours des échelles s.

• **Fonction de poids modifiée :** Le mélange spatial de patch se réalise localement comme une combinaison linéaire des patchs qui auraient pu se chevaucher si l'ordre de recollement des patchs avait été légèrement différent. On peut montrer qu'avec ce nouvel algorithme, la pondération $w_{(p,q)}$ de chaque patch (que l'on retrouve aussi dans l'Eq.(1)) dépend maintenant uniquement de la distance du point au centre des patchs les plus proches, plutôt que la distance aux patchs proprements dits (comme cela était le cas pour la méthode de fondu précédente, section 2.2).



(a) Poids utilisés dans [1].

(b) Poids utilisés avec notre méthode.

Figure 4 – *Differences entre les fonctions de poids gaussiens utilisées pour le fondu de patch entre les deux algorithmes présentés.*

C'est principalement grâce à cette approximation que l'on peut formuler une version optimisée de notre algorithme de fondu. D'un point de vue expérimental, on a pu remarquer qu'il était très difficile de faire visuellement la différence entre l'utilisation de ces deux fonctions de poids différentes.

• **Fondu spatial de patch extérieur au masque :** L'Algorithme 1 réalise un fondu spatial de patch qui s'étend naturellement en dehors du masque d'inpainting M. D'un point de vue visuel, c'est particulièrement intéressant, puisqu'on crée une transition douce entre les parties originales et reconstruites de l'image. Tous les résultats présentés en section suivante bénéficient de cette propriété. A noter qu'il est trivial de contraindre les pixels extérieurs au masque à ne pas être modifiés (en recopiant les pixels connus de I sur l'image résultat J en dernier lieu).

• **Amélioration des performances :** Nous illustrons le gain de performances de notre nouvelle approche avec l'approche précédente (section 2), ainsi qu'avec les approches de [4] (inpainting seul, sans fondu) et de Photoshop (très rapide, basée sur PatchMatch [9, 10]). La Fig.5 illustre les différences en temps de calcul sur 5 images différentes. Le temps de calcul est bien sûr proportionnel à la taille du masque de la zone à reconstruire dans chaque image.

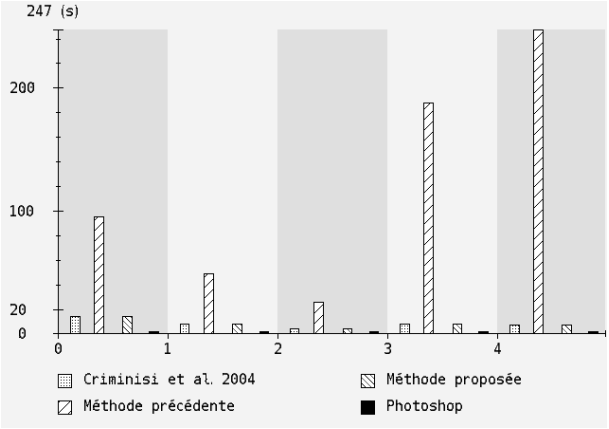


Figure 5 – *Illustration du gain en temps d'exécution entre les deux méthodes d'inpainting + fondu, présentés dans ce papier, et comparaison avec des techniques sans mélange spatial de patchs.*

On s'apercoit que le temps de calcul de notre nouvel algorithme de fondu spatial de patch devient négligeable par rapport au processus d'inpainting proprement dit (ce qui était loin d'être le cas précédemment). L'algorithme disponible en standard dans Photoshop est également plus rapide en comparaison, mais utilise probablement des optimisations matérielles poussées (GPU, multi-coeur), ce qui n'est pas notre cas (implémentation C++ standard, mono-coeur). Notons également que l'algorithme de Photoshop ne permet pas de générer des résultats avec du fondu spatial de patchs étendu sur une large portion d'image.

4 Résultats et reproductibilité

La Fig.6 illustre quelques résultats d'inpainting + fondu de patch, ainsi que des comparaisons avec des méthodes de l'état de l'art. On voit bien l'apport de l'étape de fondu de patch, qui permet de minimiser la perception des artefacts visuels, même si le recollement des patchs réalisé pendant le processus d'inpainting n'est pas parfait.

Dans un souci de reproductibilité de la recherche, nous avons réalisé un travail important pour permettre à la communauté de pouvoir réutiliser facilement cet algorithme :

• Le code source de l'algorithme (en C++) est disponible, comme une fonction `inpaint_patch()` dans les fichiers sources du projet G'MIC [11].

• Un filtre dédié a été ajouté dans le plug-in G'MIC pour GIMP [1], permettant de lancer notre algorithme à partir d'une interface graphique simple d'utilisation, sous le logiciel libre de retouche d'image GIMP. G'MIC définit également une interface en ligne de commande, permettant de lancer cet algorithme sur un ensemble d'images avec un jeu de paramètres donnés.

En conclusion, nous avons proposé un algorithme efficace et réutilisable, permettant de minimiser les artefacts visuels d'inpainting par une technique de fondu spatial de patch. Notre méthode est clairement intéressante tant elle améliore de façon significative les résultats d'inpainting obtenus (en nous comparant avec des méthodes de l'état de l'art), tout en demandant un surcoût algorithmique quasiment négligeable.

Références

[1] M. Daisy, D. Tschumperlé, et O. Lézoray. Spatial patch blending for artefact reduction in pattern-based inpainting techniques. Dans *Int. Conf. on Computer Analysis of Images and Patterns(CAIP)*, pages 523–530, York, UK, 2013.

[2] David Tschumperlé et Rachid Deriche. Vector-valued image regularization with pdes : A common framework for different applications. *IEEE Trans. PAMI*, 27(4) :506–517, 2005.

[3] M. Bertalmio, G. Sapiro, V. Caselles, et C. Ballester. Image inpainting. Dans *Proc. of the 27th annual SIGGRAPH conference*, SIGGRAPH '00, pages 417–424, New York, NY, USA, 2000.

[4] A. Criminisi, P. Pérez, et K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Trans. Im. Proc.*, 13(9) :1200–1212, Septembre 2004.

[5] O. Le Meur, J. Gautier, et C. Guillemot. Examplar-based inpainting based on local geometry. Dans *ICIP*, pages 3401–3404, Brussel, Belgium, 2011.

[6] P. Arias, G. Facciolo, V. Caselles, et G. Sapiro. A variational framework for exemplar-based image inpainting. *Int. J. Comput. Vision*, 93(3) :319–347, Juillet 2011.

[7] N. Kawai, T. Sato, et N. Yokoya. Image inpainting considering brightness change and spatial locality of textures and its evaluation. Dans *Proc. of the 3rd PSIVT*, PSIVT '09, pages 271–282, Berlin, Heidelberg, 2009.

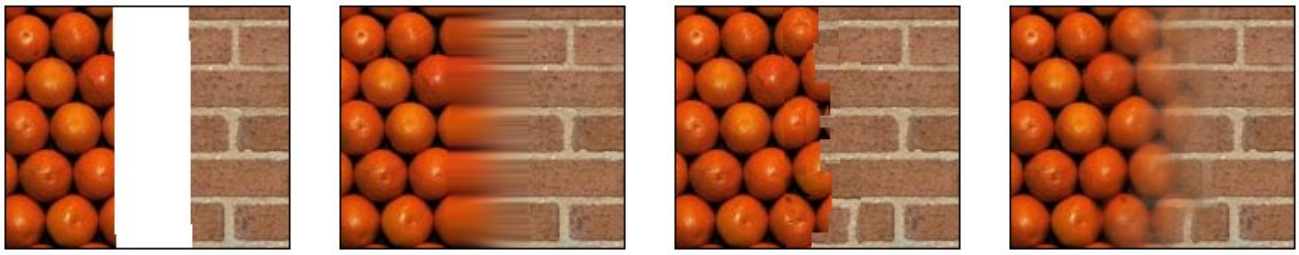
[8] J. Sun, L. Yuan, J. Jia, et H-Y. Shum. Image completion with structure propagation. *ACM Trans. Graph.*, 24(3) :861–868, Juillet 2005.

[9] Connelly Barnes, Eli Shechtman, Adam Finkelstein, et Dan B Goldman. Patchmatch : a randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3) :24 :1–24 :11, Juillet 2009.

[10] Y. Wexler, E. Shechtman, et M. Irani. Space-time completion of video. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(3) :463–476, Mars 2007.

[11] David Tschumperlé. G´MIC : GREYC´s Magic for Image Computing. *http ://gmic.sourceforge.net*, 2013.

1. http ://www.gimp.org



(a) Image couleur avec zone à reconstituer.

(b) Inpainting obtenu avec EDP de diffusion anisotrope [2].

(c) Inpainting obtenu avec méthode par patch [4].

(d) Inpainting + Fondu de patch (notre méthode).

(e) Image couleur avec zone à reconstituer.

(f) Inpainting obtenu avec méthode par patch [4].

(g) Inpainting obtenu avec l'algorithme de Photoshop [9, 10]

(h) Inpainting + Fondu de patch (notre méthode).

(i) Image couleur avec zone à reconstituer.

(j) Inpainting obtenu avec méthode par patch [4].

(k) Inpainting obtenu avec l'algorithme de Photoshop [9, 10]

(l) Inpainting + Fondu de patch (notre méthode).

(m) Image couleur avec zone à reconstituer.

(n) Inpainting obtenu avec méthode par patch [4].

(o) Inpainting obtenu avec l'algorithme de Photoshop [9, 10]

(p) Inpainting + Fondu de patch (notre méthode).

Figure 6 – *Quelques exemples d'inpainting et comparaisons avec méthodes de l'état de l'art.*