Inpainting d'Images Couleurs par Lissage Anisotrope et Synthèse de Textures
Color Image Inpainting by Anisotropic Smoothing and Texture Synthesis

V. Do G. Lebrun L. Malapert C. Smet D. Tschumperlé

Equipe Image / Laboratoire GREYC (UMR CNRS 6072), 6 Bd du Maréchal Juin, 14050 Caen CEDEX.
David.Tschumperle@greyc.ensicaen.fr

Résumé

Nous nous intéressons au problème de la reconstruction de données manquantes dans une image couleur contenant de la texture. Ce processus, couramment nommé "inpainting", nécessite une interpolation spatiale intelligente des pixels connus de l'image. Dans cet article, nous proposons de réaliser cette interpolation en deux étapes distinctes : tout d'abord, nous reconstruisons les isophotes dans les régions de données manquantes en utilisant un processus de lissage anisotrope par EDP multi-valuée, qui permet de bien recomposer la géométrie globale des structures de l'image. Dans un deuxième temps, nous re-synthétisons l'information texture à l'intérieur de ces régions par un algorithme spécifique basé sur une mise en correspondance de blocs images. Notre technique d'inpainting est illustrée avec plusieurs exemples de traitements sur des images couleurs.

Mots Clés

Inpainting, interpolation, EDP de diffusion anisotrope, synthèse de texture, images couleurs.

Abstract

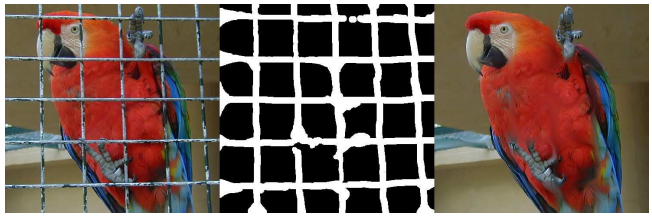
We are interested in the reconstruction of missing data in color images, by the means of spatial interpolations that preserve textures. This so-called "inpainting" process has several applications in the field of image processing. We propose a two-step algorithm for this purpose : first, we reconstruct the image isophotes in missing data regions using multi-valued PDE's that perform anisotropic smoothing. Then, we synthetize the missing textures therein using a smart bloc matching scheme. Finally, we illustrate our algorithm for real objects removal in color photographs.

Keywords

Inpainting, interpolation, anisotropic diffusion PDE, texture synthesis, color images.

1 Introduction

C'est assez récemment que le concept d' "inpainting" est apparu dans le domaine du traitement d'image numérique. De manière générique, ce terme désigne le fait de déterminer, de la manière la plus automatique possible, la couleur de pixels considérés comme *manquants* dans une image, c'est à dire, dont on ne connait pas les valeurs à priori. Ce type d'algorithme de reconstruction de données est intéressant à plus d'un titre, de par les nombreuses application concrètes qu'il peut traiter. Un algorithme d'inpainting peut être utilisé par exemple pour restaurer de manière numérique des images dégradées par des artefacts ayant détruit de manière complète certaines parties des images (rayures sur des films anciens ou taches sur des photographies). Une application dérivée intéressante consiste en la *suppression cohérente* d'objets réels dans des images, en signifiant à l'algorithme que ces objets sont considérés comme des artefacts que l'on veut corriger. Pour fonctionner, un algorithme d'inpainting nécessite donc, d'une part la donnée de l'image $\mathbf{I}$ à traiter, et d'autre part, un masque binaire M définissant les zones que l'on cherche à reconstruire. Un exemple d'utilisation pour la suppression d'objets réels est illustré en Fig.1 (l'algorithme étant celui proposé dans nos précédents travaux [28, 30]).



De gauche à droite : Image couleur $\mathbf{I}$, Masque binaire M, et résultat de l'inpainting.

FIG. 1 – Illustration d'un algorithme d'inpainting pour la suppression d'un objet dans une image couleur.

Cette grande variété d'application explique l'engouement important pour ce type de méthodes, rencontré récemment dans la littérature relative au traitement d'images. On peut tout d'abord citer les travaux précurseurs de Masnou et Morel [20], qui ont été rapidement suivis par une classe d'approches se basant sur les méthodes variationnelles et les EDP de diffusion [4, 6, 12, 27, 28, 30]. Ces méthodes se basent sur les propriétés de lissage non-linéaire des EDP pour diffuser, dans les régions à reconstruire, les valeurs des pixels voisins de ces régions. Leur intérêt principal réside dans leur capacité à reconstruire des données images ayant des géométries complexes (structures non linéaires) à l'intérieur des régions de données manquantes.
Cependant, ces méthodes consistent en des lissages ou des transports locaux orientés des intensités des pixels et sont

donc incapables de reconstruire des régions texturées, laissant parfois une impression d'aplat non réaliste sur les images traitées qui contiennent initialement beaucoup de textures.

La prise en compte de la texture dans les algorithmes d'inpainting a été envisagée récemment sous plusieurs angles. Un formalisme EDP d'inpainting prenant en compte une modélisation des textures a été proposé dans [7], se basant sur une décomposition de l'image dans les espaces BV (fonctions à variations bornées) et G (fonctions oscillantes [21]). Malheureusement, les textures de grande échelle (macro-textures) sont difficilement reconstruites par ce type d'approche. A l'inverse, dans [2, 14, 16, 32, 35], des algorithmes de synthèse de textures, basées sur des mises en correspondance et des recopies de blocs images sont proposés. Les grandes textures sont alors bien recomposées, mais on perd la propriété de reconstruction d'une géométrie globale cohérente à l'intérieur des régions de données manquantes, notamment quand la région à reconstruire recouvre deux zones de textures différentes. De plus, ces algorithmes (excepté [14]) nécessitent la connaissance d'une texture "modèle" à appliquer, qui n'est pas connue dans le cas général de l'inpainting, où l'image donnée en entrée est constitué en général d'un panel de textures aux caractéristiques très différentes.

Dans cet article, nous proposons une méthode d'inpainting simple à mettre en oeuvre, qui permet à la fois de bénéficier de la reconstruction d'une géométrie globale cohérente (même pour les grandes régions manquantes), mais aussi de re-synthétiser de la texture à l'intérieur de ces régions. Cet algorithme fonctionne en deux étapes distinctes :

• Une première étape, de reconstruction de la géométrie globale, à l'intérieur des régions de données manquantes : cette étape se base naturellement sur une méthode de type EDP de diffusion (avec contrainte de courbure), et permet la reconstruction de données images *homogènes, séparées par des discontinuités* à l'intérieur des régions d'inpainting. (section 2).

• Une deuxième étape, de synthèse des textures à l'intérieur des régions de données manquantes : Cette étape se base sur les techniques récentes de synthèse de textures utilisant des correspondances de bloc images [2, 32]. La nouveauté ici est de prendre en compte le résultat de la première étape pour synthétiser les textures à l'intérieur des régions *tout en s'adaptant à la géométrie globale déjà reconstruite*. Nous proposons pour cela une méthode simple de sélection automatique de textures modèles à considérer pour cette phase de synthèse (section 3 et 4).

Après un rapide résumé des méthodes déjà existantes, nous détaillerons comment celles-ci peuvent se spécialiser pour s'adapter à notre problème d'inpainting d'images couleurs texturées. Notre algorithme en deux étapes sera finalement illustré avec quelques résultats d'applications d'inpainting sur des images couleurs (section 5).

2 Reconstruction de la géométrie globale des régions manquantes

2.1 Inpainting et EDP de diffusion

L'inpainting peut être vu comme une interpolation intelligente des pixels de l'image bordant les régions à reconstruire, dans le but d'estimer des valeurs de pixel cohérentes à l'intérieur de ces régions. Le formalisme des EDP de diffusion anisotrope est particulièrement bien adapté pour de telles interpolations. Ce type d'EDP a été principalement utilisé pour la régularisation d'images, depuis les travaux précurseurs de Perona-Malik [22], et de nombreuses variantes d'EDP de diffusion ont été proposées depuis lors (par exemple dans [1, 3, 8, 10, 13, 19, 23, 25, 26, 33], cette liste étant loin d'être exhaustive). Leur principe repose sur l'application de diffusions successives localement orientées des valeurs des pixels de l'image, agissant ainsi comme des filtres moyenneurs adaptés aux structures présentes dans l'image. Dans le cadre de l'inpainting, cette possibilité de diffuser les valeurs des pixels *à l'intérieur des régions à reconstruire* est très intéressante et a été utilisée avec succès dans [4, 6, 12, 30].
Nous adaptons ici nos travaux précédents [27, 28, 30], où nous avions défini des EDP de lissage anisotrope génériques qui permettent de bien contrôler l'orientation et la force du lissage local effectué. Dans notre cas, nous voulons compléter de la manière la plus cohérente possible les isophotes à l'intérieur des régions de données manquantes. Soit $\mathbf{I} : \Omega \rightarrow \mathbb{R}^3$ une image couleur. Les composantes Rouge,Vert,Bleu de cette image seront dénotés par I_1,I_2,I_3. $M : \Omega \rightarrow \{0,1\}$ désigne le masque d'inpainting, où les régions définies par $M(x,y) = 1$ sont considérées comme inconnues, et donc à reconstruire. Nous proposons l'EDP de lissage anisotrope à préservation de courbure suivante (mentionnée dans [28, 29]) :

$$\frac{\partial I_i}{\partial t} = M_{(x,y)} \ \left(\text{Trace}(\mathbf{u}\mathbf{u}^T\mathbf{H}_i) + \nabla I_i^T \mathbf{J}_\mathbf{u}\mathbf{u}\right) \qquad (1)$$

où $\mathbf{H}_i$ est la matrice Hessienne de I_i (matrice des dérivées secondes), et $\mathbf{u}$ correspond à l'estimation de la direction de l'isophote en chaque point, et est calculé comme le vecteur propre correspondant à la valeur propre la plus faible *du tenseur de structure lissé* $\mathbf{G}_\sigma = (\sum_{k=0}^{3} \nabla I_k \nabla I_k^T) * G_\sigma$, comme défini dans [15, 33]. $\mathbf{J}_\mathbf{u}$ représente alors la matrice jacobienne de $\mathbf{u}$ (matrice des dérivées premières).
L'équation (1) a plusieurs propriétés intéressantes :

- Notons tout d'abord que (1) ne modifie l'image $\mathbf{I}$ que dans les régions à "inpainter", c-à-d où $M_{(x,y)} = 1$.
- Au fur et à mesure des itération de l'EDP (1), les pixels voisins aux régions définies par $M_{(x,y)} = 1$ diffusent à l'intérieur de ces régions jusqu'à les remplir complètement. Ceci est fait en chaque point dans la direction des isophotes $\mathbf{u}$, qui est elle-même ré-évaluée itération par itération.
- Le terme de contrainte $\nabla I_i^T \mathbf{J}_\mathbf{u}\mathbf{u}$ dans (1) permet la préservation des courbures (voir [28] pour plus

de détails). En particulier, il permet de rejoindre des isophotes se trouvant de part et d'autre de la région à reconstruire, avec des courbes autres que des droites. D'un point de vue géométrique, il contraint l'équation de la chaleur mono-dimensionnelle orientée $\frac{\partial I_i}{\partial t} = \text{Trace}(\mathbf{u}\mathbf{u}^T\mathbf{H}_i)$ (premier terme de l'EDP (1)) sur la courbe intégrale du champ de vecteur $\mathbf{u} : \Omega \rightarrow \mathbb{R}^2$, passant par le point (x, y).

Intuitivement, on peut voir l'application de notre EDP de diffusion (1) comme la propagation des informations de couleur de l'extérieur des zone à inpainter vers l'intérieur de ces zones, sans pour autant produire une région de couleur uniforme qui aurait la couleur moyenne des pixels entourant la région. Au contraire, l'EDP (1) permet de reconstruire des géométries comportant plusieurs régions de couleurs différentes, séparées par des discontinuités relativement nettes. Cette propriété est illustrée en Fig.2 qui est un agran-dissement de la Fig.1. La géométrie globale de l'image reconstruite par l'EDP de diffusion (1) est relativement complexe.

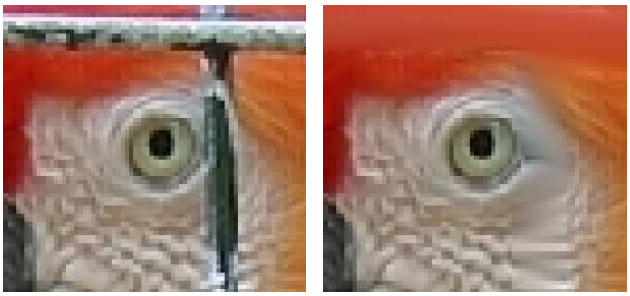


FIG. 2 – Résultat d'inpainting par l'EDP de diffusion (1) (détail).

Par contre, sur des images fortement texturées, les limitations de ce type d'algorithmes apparaissent nettement (voir l'exemple de la Fig.9 en fin d'article). L'inpainting par EDP de diffusion anisotrope permet de *préserver et de compléter les discontinuités* existantes dans les images, mais en aucun cas *n'en crée de nouvelles*. Aucune texture ne peut être recomposée à l'intérieur des régions d'inpainting ainsi interpolées. Une seconde étape est donc nécéssaire pour reconstruire l'information texturée.

3 Synthèse de Textures

3.1 Synthèse à partir d'une texture modèle

Nous résumons ici deux algorithmes de synthèse de textures [2, 32] récemment proposés dans la littérature. L'idée est de créer, à partir d'une petite image de texture (appelée *texture modèle*), une nouvelle image, plus grande, qui semble obtenue par le même processus stochastique que celui qui aurait servi à générer le modèle. Ces algorithmes prennent comme entrées le modèle de texture I_e d'une part, et une image I_s contenant du bruit blanc où sera synthétisée la texture désirée d'aute part. Le parcours de l'image destination I_s se fait de façon linéaire (de haut en bas, de gauche à droite). On peut noter que les pixels bruités servent uniquement à l'initialisation du processus, lors du parcours de la première ligne (Fig.3).

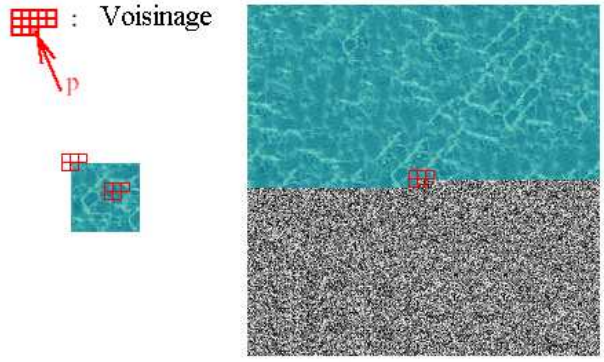


FIG. 3 – Synthèse de texture par l'algorithme [32]. A gauche, la texture modèle et à droite, l'image en cours de synthèse.

Algorithme de Wei-Levoy : Pour déterminer la valeur d'un pixel $p \in I_s$, son voisinage $L_{s(p)}$ (en forme de L renversé, comme le montre la Fig.3) est comparé à tous les voisinages possibles $L_{e(p')}$ de même taille et de même forme contenus dans la texture d'entrée I_e. Le pixel de la texture d'entrée ayant le voisinage le plus proche (au sens de la norme L_2) donne sa valeur au pixel de l'image de sortie :

$$I_s(p) = I_e(p') \qquad \text{où} \quad p' = \text{argmin}_l \|L_{s(p)} - L_{e(l)}\|^2$$

Ce procédé est répété pour chaque pixel p de l'image de sortie. Dans l'algorithme initial, le voisinage considéré a une forme de L renversé car on ne considére dans le calcul de similarité que les pixels ayant été déjà synthétisés. Sa taille a également une importance dans la qualité de la texture obtenue. Intuitivement, elle correspond à l'échelle de la plus large structure de la texture d'entrée (aussi appelée *texel*). L'algorithme peut ainsi être écrit en pseudo-code de la manière suivante :

```
Function SyntheseTexture(I_e , I_s)
1  Pour chaque pixel p de I_s
2      Pour chaque pixel p' de I_e (texture d'entrée)
3      Comparer les deux voisinages L_(e) et L_(s).
4      Si le voisinage L_(e) est le plus ressemblant, sauver
la couleur du pixel dans p_s.
5      Fin boucle
6      Donner à I_s(p) sa valeur I_e(p').
7  Fin boucle
```

Notons que chaque pixel p généré nécéssite la comparaison de son voisinage $L_{(p)}$ avec tous les voisinages existants dans la texture modèle I_e, ce qui est un processus extrêmement coûteux. Pour une image de départ de 50x50

et une image de sortie 256x256, la synthèse nécéssite environ 3 heures (sur une station SUN Solaris 1.2 Ghz) (Fig.4).

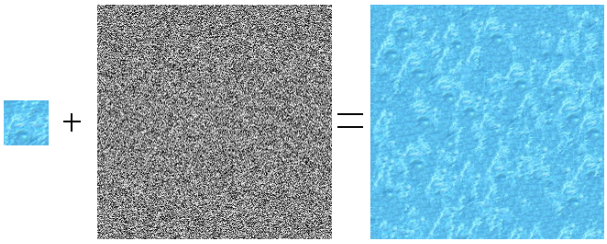


FIG. 4 – Résultats obtenus par [32] sur une texture d'eau.

Pour éviter cette recherche exhaustive et optimiser le calcul des pixels optimum, Ashikhmin a proposé dans [2] une variante plus rapide de l'algorithme de Wei-Levoy.

Algorithme d'Ashikhmin : Ashikhmin part de l'observation que lors du processus de synthèse de l'image, on a déjà trouvé des pixels dans la texture d'entrée ayant des voisinages similaires au voisinage décalé du point courant dans l'image de sortie. Cette information n'est pas utilisée dans l'algorithme original où nous re-parcourons toute l'image de texture d'entrée pour resynthétiser chaque nouveau pixel. Il est raisonnable de penser que le meilleur candidat dans la texture d'entrée se trouvera probablement dans le voisinage décalé des pixels déjà calculés précédemment (Fig.5). On réduit ainsi de manière significative l'espace de recherche des pixels optimaux, et l'algorithme s'exécute alors beaucoup plus rapidement. Cet algorithme optimisé peut s'écrire comme :

Function SyntheseTextureRapide(I_e , I_s)
1 Initialiser le tableau des positions originales des pixels avec des positions valides aléatoires.
2 Pour chaque pixel p de I_s
3 Pour chaque pixel du voisinage de p
4 Utiliser sa position originale stockée dans le tableau pour générer un candidat.
5 Fin boucle
6 Chercher le candidat ayant le maximum de vraisemblance (au sens de la norme L_2).
7 Sauver la nouvelle position originale dans le tableau.
8 Donner à $I_e(p)$ sa valeur optimale.
9 Fin boucle

Cette approche tend à créer une texture de sortie comme un *collage de différents morceaux continus de la texture modèle originale*. La qualité de la texture synthétisée est moins bonne, car la recherche des pixels les plus cohérents n'est pas exhaustive (Fig.6). Néanmoins, elle reste généralement de qualité suffisante pour la synthèse de textures de tailles moyennes, et c'est sur cette base que nous allons élaborer notre phase de texturage.

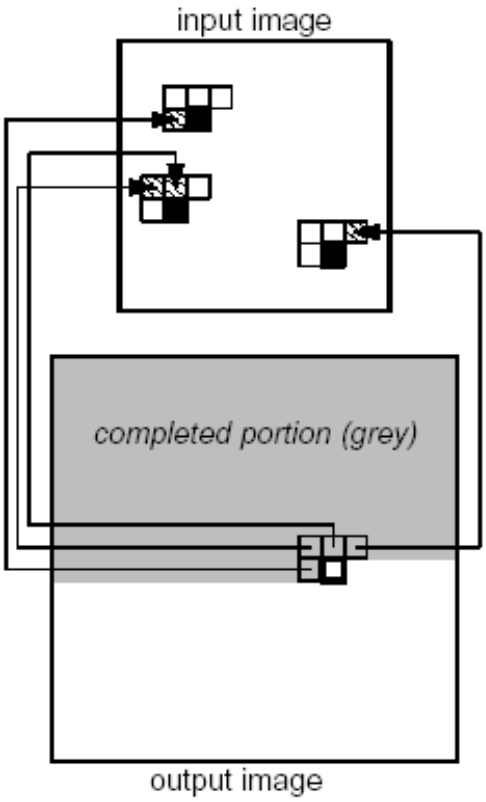


FIG. 5 – Pixels candidats pour l'algorithme d'Ashikhmin. Chaque pixel du voisinage en forme de L génère un candidat selon sa position originale dans la texture d'entrée (figure extraite de [2]).

4 Adaptations pour l'inpainting.

Pour utiliser l'algorithme de synthèse de texture [2] dans le cadre de l'inpainting d'images couleurs texturées, nous proposons plusieurs adaptations, décrites ci-après.

4.1 Forme du voisinage et sens de parcours

Dans les algorithmes de Wei-Levoy [32] et Ashikhmin [2], les voisinages utilisés pour les comparaisons ont une forme de L, afin de considérer seulement les pixels déjà synthétisés dans ces voisinages. Dans notre cas, les régions à texturer ont des formes quelconques et contiennent même des pixels dont *la couleur est significative* puisqu'une première phase d'interpolation de la géométrie a déjà été réalisée par l'EDP de diffusion anisotrope (1).
Ici, nous pouvons donc utiliser naturellement un masque complet de texturage, de forme carré et non plus en forme de L renversé. Ceci nous permet ainsi de prendre en compte à la fois les pixels déjà calculés par l'algorithme de texturage, les pixels calculés par la première phase d'inpainting et ceux de l'image de départ.
Le sens de parcours pour la synthèse de texture est également transformé : les pixels appartenant aux bords des régions à inpainter sont traités de manière prioritaire. Ainsi, on définit une sorte d'*érosion texturée*, qui reconstruit au

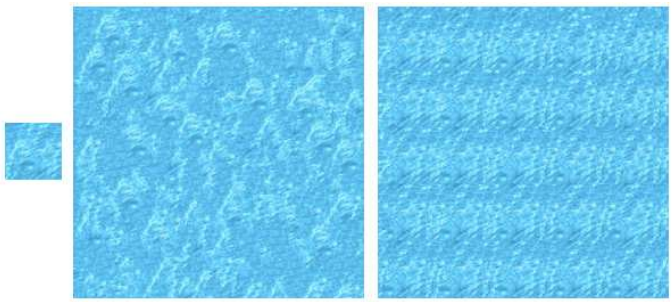


FIG. 6 – Résultats comparés obtenus avec la méthode classique et la méthode rapide.

fur et à mesure les pixels de l'extérieur vers l'intérieur des régions. Les pixels des bords étant les plus proches des "vrais" pixels de l'image, les nouveaux pixels synthétisés utilisent le maximum d'informations cohérentes de leur voisinage et permettent ainsi une transition plus souple entre la région à interpoler et le reste de l'image.

4.2 Sélection automatique de textures

Nous avons vu que l'algorithme de synthèse de texture dans les zones de données manquantes par notre algorithme modifié d'Ashikhmin nécéssite la donnée d'une texture modèle. Bien sûr, une idée naturelle serait d'utiliser toute l'image d'entrée (hors régions inconnues) comme texture modèle. Mais cette solution intuitive n'est pas intéressante : la recherche des voisinages proches pour la synthèse de texture utilise en effet une norme L_2 comme critère de similarité, et il arrive que des blocs visuellement très différents mais étant proches au sens de la norme L_2 soient sélectionnés. La recherche d'un meilleur critère de similarité entre deux images pourrait être envisagé. C'est en réalité un problème très complexe, et la recherche d'une norme plus adaptée n'est donc pas une piste encourageante. En nous inspirant de l'idée d'Ashikhmin, nous proposons plutôt de *réduire l'espace de recherche*, c-à-d de définir la texture modèle comme *une sous-partie segmentée de l'image originale*. Notons que cette segmentation n'a pas besoin d'être très précise, car on cherche juste à limiter l'espace de recherche de manière cohérente, et pas forcément optimale : par la suite, le critère de similarité utilisé pour la synthèse se chargera d'éliminer les pixels mal segmentés. Dans cet optique, une méthode de segmentation simple, basée sur un algorithme de croissance de région, est donc bien adaptée ici.

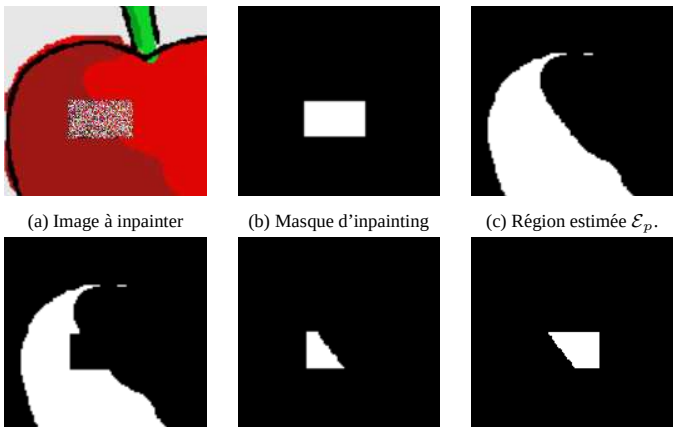
Notre algorithme doit donc d'une part, décomposer la zone à reconstruire en plusieurs parties ayant vraissemblablement des textures différentes, et d'autre part, pour chacune de ces zones, trouver la texture modèle correspondante dans l'image $\mathbf{I}$ qui va servir dans l'algorithme de synthèse de texture. Pour cela, nous proposons l'algorithme suivant. Il est illustré sur un cas synthétique, à titre d'exemple (Fig.7).

1 Tant que tout le masque d'inpainting n'est pas couvert.
2 Chercher un point p du masque sur la frontière.
3 Effectuer une croissance de région à partir de p sur l'image $\mathbf{I}^* = \mathbf{I} * G_\sigma$, pour définir un ensemble de pixels $\mathcal{E}_p$ d'intensité "homogène" (Fig.7c). Ici, G_σ représente un filtre de lissage gaussien de dimension 2.
4 Séparer cette régions en deux régions $\mathcal{E}_{p1}$ et $\mathcal{E}_{p2}$ où $\mathcal{E}_{p1} \cup \mathcal{E}_{p2} = \mathcal{E}_p$, et

$$\begin{cases} \forall p^{'} \in \mathcal{E}_{p1}, \quad M(p^{'}) = 0 \\ \forall p^{'} \in \mathcal{E}_{p2}, \quad M(p^{'}) = 1 \end{cases}$$

(Fig.7d,e).
5 Appliquer la synthèse de texture dans $\mathcal{E}_{p2}$ avec la texture modèle $\mathcal{E}_{p1}$.
6 Ré-initialiser à $M(p) = 0$ tous les points de $\mathcal{E}_{p2}$ qui ont été synthétisés par l'étape 5. Le masque résiduel est alors utilisé pour l'itération suivante de l'algorithme (Fig.7f).
7 Fin boucle.

Notons dans l'étape 3, que la segmentation par croissance de régions est d'une part effectuée sur une version lissée $\mathbf{I}^*$ de l'image $\mathbf{I}$, et d'autre part qu'elle utilise un critère de similarité pour distinguer les pixels d'une même région de type $S(p_1, p_2) = \|\mathbf{I}^*(p_1) - \mathbf{I}^*(p_2)\|^2$. On doit alors avoir $S(p_1, p_2) < \tau$ où τ est un seuil défini par l'utilisateur, pour considérer que p_1 et p_2 appartiennent à la même région. L'écart type σ du filtre de lissage gaussien est bien sûr proportionnel à l'échelle des textures considérées.



(a) Image à inpainter (b) Masque d'inpainting (c) Région estimée $\mathcal{E}_p$.
(d) Région $\mathcal{E}_{p1}$ (d) Région $\mathcal{E}_{p2}$ (d) Masque résiduel.

FIG. 7 – Illustration de la première itération de notre méthode de sélection automatique de textures modèles.

Cette méthode de sélection automatique, bien que très simple dans son principe, permet de bien limiter l'espace de recherche (le domaine de définition des textures modèles) pour la phase de synthèse de texture. Cette deuxième phase permet une reconstruction correcte des textures dans les régions à "inpainter".

5 Applications

Nous avons appliqué notre algorithme d'inpainting recréant les textures dans le cadre de la suppression d'éléments réels présents dans des photographies couleur. Les résultats sont présentés à la fin de cet article, sur les Fig.8,9,10. Les temps d'exécution sont de l'ordre de quelques minutes, sur une SUN SPARC 1.2Ghz tournant sous Solaris 9. La première phase de l'algorithme, basée sur les EDP de diffusion, est celle qui nécessite le plus de calculs. L'algorithme a été programmé en C++, utilisant la bibliothèque de traitement d'image générique *CImg* (`http://cimg.sourceforge.net/`)[31].

On peut remarquer en particulier les propriétés suivantes :

- Sur la Fig.9, l'intérêt de la phase de synthèse de textures est mis en évidence. L'image est ici fortement texturée et l'application d'une méthode d'inpainting non-texturé introduit des aplats très visibles sur l'image reconstruite (Fig.9b, ici en utilisant la méthode décrite dans [28]). Notre méthode à préservation de textures permet ici de reconstruire une image cohérente (Fig.10).
- Les Fig.8 et 9a,b montrent également la propriété de reconstruction de la géométrie globale des isophotes. Les frontières entre les différentes parties texturées sont bien réliées entre elle, malgré la taille importante des régions à reconstruire. Ceci est possible grâce à l'utilisation de notre EDP de diffusion (1), dans la première phase de reconstruction des isophotes de notre algorithme.

Conclusion & Perspectives

Dans cet article, nous avons proposé un algorithme d'inpainting original et très simple, en deux étapes distinctes qui permet dans un premier temps de reconstruire la géométrie globale des isophotes de l'image, puis dans un second temps, de re-synthétiser l'information texture à l'intérieur des régions occultées, en prenant en compte la géométrie reconstruite lors de la première phase. Pour chaque étape, nous nous sommes basés sur l'état de l'art des méthodes existantes de diffusion anisotropes par EDP et de synthèse de textures. Ces méthodes ont été adaptées pour répondre aux spécificités de l'inpainting, notamment avec l'élaboration d'un algorithme de sélection automatique de textures modèles simple mais efficace. Les résultats obtenus sont très encourageants, et nous projetons d'étendre notre méthode pour agir sur des données vidéos, par exemple pour supprimer des sous-titres ou des logos dans des films de manière cohérente.

Références

[1] L. Alvarez, F. Guichard, P.L. Lions, and J.M. Morel. *Axioms and fundamental equations of image processing.* Archive for Rational Mechanics and Analysis, 123(3) :199–257, 1993.

[2] M. Ashikhmin. *Synthesizing Natural Textures.* Proceedings of 2001 ACM Symposium on Interactive 3D Graphics, Research Triangle Park, NorthCarolina March 19-21, pp. 217-226

[3] G. Aubert and P. Kornprobst. *Mathematical Problems in Image Processing : Partial Differential Equations and the Calculus of Variations*, vol.147 of Applied Mathematical Sciences. Springer-Verlag, January 2002.

[4] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera. *Filling-in by joint interpolation of vector fields and grey levels.* University of Minnesota IMA TR, April 2000.

[5] M. Bertalmio, L.T. Cheng, S. Osher, G. Sapiro. *Variational Problems and Partial Differential Equations on Implicit Surfaces.* Computing and Visualization in Science, 3(3) :159–167, 2000.

[6] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. *Image inpainting.* Proceedings of the SIGGRAPH, p.417–424. ACM Press, Addison Wesley Longman, 2000.

[7] M. Bertalmio, L. Vese, G. Sapiro and S. Osher. *Simultaneous Structure and Texture Image Inpainting.* Proceedings of CVPR'03, Madison/Wisconsin, June 16-22, 2003.

[8] M.J. Black, G. Sapiro, D.H. Marimont, and D. Heeger. *Robust anisotropic diffusion.* IEEE Transaction on Image Processing, 7(3) :421–432, 1998.

[9] V. Caselles, J.Morel and C. Sbert. *An Axiomatic Approach to Image Interpolation.* IEEE Transaction on Image Processing, 7 :376-386, 1998.

[10] A. Chambolle and P.L. Lions. *Image recovery via total variation minimization and related problems.* Nümerische Mathematik, 76(2) :167–188, 1997.

[11] T. Chan and J. Shen. *Variational restoration of non-flat image features : Models and algorithms.* Research Report. Computational and applied mathematics department of mathematics Los Angeles, June 1999.

[12] T. Chan and J. Shen. *Non-texture inpaintings by curvature-driven diffusions.* Technical Report 00-35, Department of Mathematics, UCLA, Los Angeles, September 2000.

[13] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud. *Deterministic edge-preserving regularization in computed imaging.* IEEE Transactions on Image Processing, 6(2) :298–311, 1997.

[14] A. Criminisi, P. Pérez, K. Toyama. *Object Removal by Exemplar-Based Inpainting.* Conference on Computer Vision and Pattern Recognition, 2003, pp 721-728.

[15] S. Di Zenzo. *A note on the gradient of a multi-image.* Computer Vision, Graphics and Image Processing, 33 :116-125, 1986.

[16] A. Efros and W.T. Freeman. *Image Quilting for Texture Synthesis and Transfer.* Proceedings of SIGGRAPH '01, Los Angeles, California, August, 2001.

[17] A. Efros and T. Leung. *Texture Synthesis by Non-Parametric Sampling.* IEEE International Conference on Computer Vision, p 229-248, 1999.

[18] J. Jia and C.-K. Tang. *Image Repairing : Robust Image Synthesis by Adaptive ND Tensor Voting.* Proceedings of CVPR'03, Madison/Wisconsin, June 16-22, 2003.

[19] R. Kimmel, R. Malladi, and N. Sochen. *Images as embedded maps and minimal surfaces : movies, color, texture, and volumetric medical images.* International Journal of Computer Vision, 39(2) :111–129, September 2000.

[20] S. Masnou and J.M. Morel. *Level-lines based disocclusion.* 5th IEEE International Conference on Image Processing, Chicago, IL. Oct 4-7, 1998.

[21] Y. Meyer. *Oscillating Patterns in Image Processing and Nonlinear Evolution.* American Mathematical Society, ISBN : 0821829203, September 2001.

[22] P. Perona and J. Malik. *Scale-space and edge detection using anisotropic diffusion.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(7) :629–639, July 1990.

[23] L. Rudin, S. Osher, and E. Fatemi. *Nonlinear total variation based noise removal algorithms.* Physica D, 60 :259–268, 1992.

[24] G. Sapiro. *Geometric Partial Differential Equations and Image Analysis*. Cambridge University Press, 2001.

[25] G. Sapiro and D.L. Ringach. *Anisotropic diffusion of multi-valued images with applications to color filtering.* IEEE Transactions on Image Processing, 5(11) :1582–1585, 1996.

[26] N. Sochen, R. Kimmel, and A.M. Bruckstein. *Diffusions and confusions in signal and image processing.* Journal of Mathematical Imaging and Vision, 14(3) :195–209, 2001.

[27] D. Tschumperlé. *PDE's Based Regularization of Multi-valued Images and Applications.* PhD Thesis, Université de Nice-Sophia Antipolis/France, December 2002.

[28] D. Tschumperlé. *Fast Anisotropic Smoothing of Multi-Valued Images using Curvature-Preserving PDE's.* Research Report "Les Cahiers du GREYC", No 05/01. Equipe IMAGE/GREYC (CNRS UMR 6072), Février 2005.

[29] D. Tschumperlé. *LIC-based Regularization of Multi-Valued Images.* To appear in IEEE International Conference on Image Processing (ICIP'05), Genova/Italy, October 2005.

[30] D. Tschumperlé, R. Deriche. *Vector-Valued Image Regularization with PDE's : A Common Framework for Different Applications.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.27, No 4, April 2005.

[31] D. Tschumperlé. *The CImg Library : The C++ Template Image Processing Library.* `http://cimg.sourceforge.net/`.

[32] L.Y. Wei, M. Levoy. *Fast Texture Synthesis using Tree-structured Vector Quantization.* Proceedings of Siggraph 2000, Computer Graphics Proceedings, ACM Press, pp 479-488, 2000.

[33] J. Weickert. *Anisotropic Diffusion in Image Processing*. Teubner-Verlag, Stuttgart, 1998.

[34] J. Weickert *Coherence-enhancing diffusion of colour images.* Proc. VII National Symposium on Pattern Recognition and Image Analysis (VII NSPRIA, Barcelona, April 21-25, 1997), Vol. 1, 239-244, 1997.

[35] Y. Xu, B. Guo and H.Y. Shum. *Chaos Mosaic : Fast and memory efficient texture synthesis*. Tech. Report MSR-TR-2000-32, Microsoft Research, 2000.

(a) Image couleur à inpainter.

(b) Image inpaintée.

FIG. 8 – Exemple d'inpainting d'image couleur.

(a) Image couleur à inpainter (b) Résultat de l'inpainting par EDP seule (c) Résultat de l'inpainting par EDP + synthèse de texture

FIG. 9 – Illustration de l'importance de la synthèse de texture pour l'inpainting d'images.

(a) Inpainting d'image couleur (b) Zoom correspondant (c) Inpainting d'image couleur

(d) Image couleur à inpainter (e) Image inpaintée

FIG. 10 – Application de notre méthode inpainting avec préservation de textures sur des images couleurs.