Vers un inpainting vidéo automatique, rapide et générique

Alasdair NEWSON[1,2], Andrés ALMANSA[2], Matthieu FRADET[1], Yann GOUSSEAU[2], Patrick PÉREZ[1]

[1]Technicolor
975 Avenue des Champs Blancs, 35570, Cesson-Sévigné, France

[2]CNRS LTCI
46 rue Barrault, 75013, Paris, France

alasdair.newson@technicolor.com, andres.almansa@telecom-paristech.fr,
matthieu.fradet@technicolor.com yann.gousseau@telecom-paristech.fr,
patrick.perez@technicolor.com

Résumé – Dans le domaine de l'inpainting vidéo, des résultats de bonne qualité présentant une cohérence globale sont difficiles à produire de manière automatique et en un temps raisonnable. Dans cet article, nous partons du travail de Wexler *et al*. pour proposer un algorithme d'inpainting vidéo efficace tout en diminuant fortement le temps d'exécution. Nous étendons tout d'abord l'algorithme PatchMatch aux données spatio-temporelles afin d'accélérer la recherche de plus proches voisins approximatifs dans l'espace des patchs. Pour ce faire, nous proposons également une solution simple et rapide au problème récurrent de la sur-régularisation due au moyennage de patchs, qui se manifeste par des résultats flous. Nous montrons sur plusieurs exemples, dont des vidéos complexes et de résolution élevée, que nous pouvons ainsi obtenir sans intervention manuelle des résultats similaires à ceux de la méthode semi-automatique de l'état-de-l'art.

Abstract – In this paper, we propose a video inpainting algorithm which produces globally coherent results in drastically reduced execution time. We build on the work by Wexler *et al*. and we accelerate the nearest neighbour search by adapting the PatchMatch algorithm to the spatio-temporal setting. We also propose a simple and fast solution to the problem of excessive blurring, due to the averaging of patches. We find that our results are similar to a semi-automatic state-of-the-art method and can be applied to high resolution videos with no user intervention.

1 Introduction

Le but de l'inpainting (ou désoccultation) est de remplacer des régions manquantes d'une image par un contenu qui soit visuellement satisfaisant [1, 2]. Une des difficultés principales de cette tâche est la restitution simultanée des structures géométriques et des textures. Par extension, l'objectif de l'inpainting vidéo est de remplir un « trou » spatio-temporel dans une vidéo, ce qui ajoute de nouvelles difficultés techniques, en particulier la reconstruction d'objets en mouvement, l'inpainting de l'avant-plan et de l'arrière-plan, ainsi que la maitrise du temps d'exécution.

La plupart des algorithmes d'inpainting vidéo reposent soit sur le traitement d'objets, soit sur le traitement de *patchs*, c'est-à-dire des petits fragments d'image. En général, les algorithmes utilisant la première approche [3, 4] segmentent la vidéo entre avant-plan en mouvement et arrière-plan, et ces deux parties sont ensuite remplies indépendamment.

La première méthode par patchs qui assure une cohérence temporelle des résultats est celle de Wexler *et al*. [5]. Il s'agit d'une méthode itérative reposant sur la résolution d'un problème d'optimisation globale. La grande dimensionalité du problème rend l'algorithme très lent, nécessitant par exemple plusieurs jours pour traiter quelques secondes de vidéo VGA. D'un autre côté, la méthode semi-automatique récente de Granados *et al*. [6] montre des résultats impressionnants sur des images d'une résolution auparavant rédhibitoire pour une tâche d'inpainting par patchs (jusqu'à 1120x754x200). Nous proposons un algorithme rapide qui donne des résultats semblables, mais sans recourir à l'interaction et avec un gain substantiel en vitesse.

2 Inpainting vidéo avec PatchMatch 3D

Notre approche s'appuie sur les travaux de Wexler *et al*. [5]. Cet algorithme permet le remplissage d'une occultation spatio-temporelle grâce à l'information venant des parties non-occultées de la vidéo. La solution est obtenue par une heuristique de minimisation d'une fonctionnelle globale reposant sur la cohérence des patchs. L'algorithme alterne entre une recherche de *plus proches voisins* (PPVs) de patchs dans la région occultée et une reconstruction de la région avec ces PPVs. Une pyramide multi-résolution spatio-temporelle est également utilisée afin d'éviter des solutions correspondant à des minima locaux.

2.1 Approche de Wexler *et al.*

Soit $\mathcal{H}$ l'occultation spatio-temporelle et $\mathcal{D}$ la région non-occultée. Soit $p = (x, y)$ une position dans la vidéo, W_p un

Images originales : “Jumping girl” (en haut) et “Beach umbrella” (en bas) | Résultat d’inpainting de [5] | Notre résultat d’inpainting

FIGURE 1: Nous obtenons des résultats similaires à ceux de [5], en réduisant le temps de recherche des PPVAs par un facteur pouvant atteindre 50.

patch centré sur p et V_p (appartenant à $\mathcal{D}$) le PPV de W_p. Wexler *et al*. utilisent des patchs couleur de taille 5x5x5. La distance $d(W_p, V_p)$ est la somme des différences au carré des composants du patch. L’algorithme de Wexler comprend trois étapes principales : la recherche des PPVs, la reconstruction de la région occultée avec ces PPVs, et l’utilisation de la pyramide spatio-temporelle. Pour accélérer la recherche des PPVs, qui est très longue, Wexler *et al*. utilisent la méthode d’Arya *et al*. [7] pour trouver des *plus proches voisins approchés* (PPVAs).

Dans [5], deux solutions sont proposées au problème de la reconstruction. La première remplace la couleur c d’un pixel $p \in \mathcal{H}$ par la moyenne pondérée suivante :

$$c = \frac{\sum_{i \in W_p} \alpha_p^i s_p^i c^i}{\sum_{i \in W_p} \alpha_p^i s_p^i} \quad avec \quad s_p^i = e^{-\frac{d(W_i, V_i)}{2\sigma^2}}, \qquad (1)$$

où c^i est la couleur de la position qui correspond à la position p dans le PPVA de W_i. Le scalaire α_p^i est un poids donné au pixel i qui dépend de sa distance à la frontière de l’occlusion. Le paramètre σ est défini comme le trois-quartile de toutes les distances $d(W_i, V_i)$.

La deuxième solution de reconstruction repose sur une estimation de c par l’algorithme *mean-shift*. Le *mean shift* est effectué dans l’espace des couleurs c^i pondérées par $\alpha_p^i s_p^i$, et ceci avec une bande passante qui est réduite au cours de l’algorithme. Cette solution évite d’avoir un résultat d’inpainting flou, mais est plus couteuse en temps que la moyenne pondérée.

2.2 Approche proposée

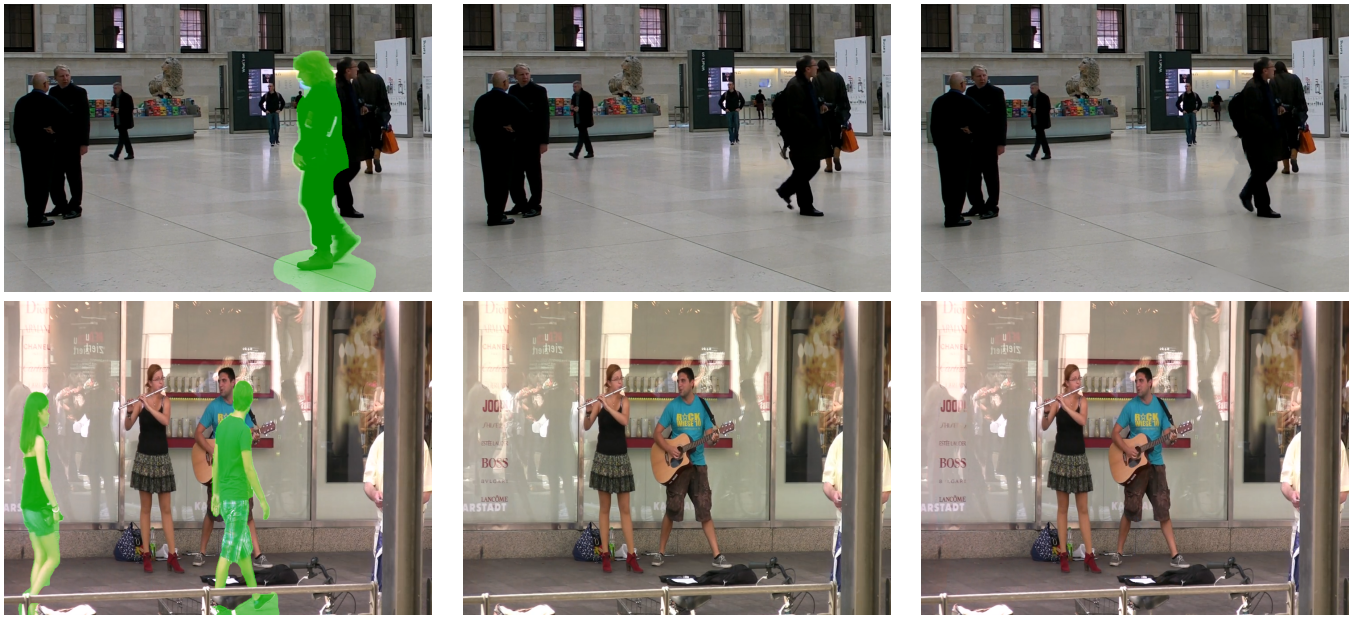
Pour accélérer la recherche de PPVAs, nous employons l’algorithme appelé « PatchMatch » [8], qui doit être étendu au cas spatio-temporel. L’utilisation de cet algorithme pour un inpainting du type [5] est suggérée dans [8] et implémentée par Darabi *et al*. dans [9] pour l’inpainting d’image, mais pas, à notre connaissance, pour la vidéo. Décrivons d’abord l’algorithme PatchMatch original. Soit $\tilde{\mathcal{D}}$ la région non-occultée dans laquelle aucun patch ne contient de pixel occulté. Nous restreignons la recherche des PPVAs à $\tilde{\mathcal{D}}$. Définissons aussi le *décalage de PPVA* de W_p comme étant le décalage spatial entre W_p et V_p. PatchMatch comporte trois étapes : l’initialisation, la propagation et la recherche aléatoire. Soit $W_{(x,y)}$ le patch 2D centré sur (x, y). L’initialisation consiste à associer des PPVAs aléatoirement à chaque W_p. Lors de la propagation, les patchs sont parcourus de manière séquentielle lexicographique. Pour un patch $W_{(x,y)}$, l’algorithme considère les décalages de PPVA des patchs $W_{(x-1,y)}$ et $W_{(x,y-1)}$. Si ces décalages de PPVA mènent à un meilleur PPVA que $V_{(x,y)}$, alors ce dernier est remplacé. L’ordre de parcours est inversé à l’itération suivante de la propagation, et l’algorithme teste les décalages de PPVA de $W_{(x+1,y)}$ et $W_{(x,y+1)}$. La troisième étape, la recherche aléatoire, consiste à chercher de meilleurs PPVAs pour chaque W_p dans une fenêtre de plus en plus petite autour de V_p. La fenêtre est définie par un cube de côté $w\alpha^j$, où α est un facteur de décroissance de la taille de fenêtre, et w est la taille maximale de la fenêtre. Cela est répété pour tous les j tels que $w\alpha^j > 1$.

2.2.1 Extension de PatchMatch au cas spatio-temporel

Nous détaillons maintenant notre extension de PatchMatch au cas 3D. Soit $W_{x,y,t}$ un patch 3D centré en (x, y, t).

Nous initialisons les PPVAs à des positions aléatoires dans $\tilde{\mathcal{D}}$. La recherche aléatoire est étendue au cas spatio-temporel de manière triviale en explorant des cubes 3D de taille décroissante autour de V_p pour chaque $p \in \mathcal{H}$.

Pour l’étape de propagation il faut définir un ordre de parcours. Nous parcourons l’axe des x, ensuite celui des y et enfin l’axe temporel. Pour chaque $W_{(x,y,t)}$, nous examinons les décalages de PPVA de $W_{(x-1,y,t)}$, $W_{(x,y-1,t)}$ et $W_{(x,y,t-1)}$ pour savoir s’ils mènent à de meilleurs PPVAs de $W_{(x,y,t)}$. Entre itérations de la propagation, l’ordre de parcours est inversé, et les décalages de PPVA de $W_{(x+1,y,t)}$, $W_{(x,y+1,t)}$ et $W_{(x,y,t+1)}$ sont testés. Pour cette étape, il est important de savoir si l’ordre de parcours est bien choisi. En particulier, les propriétés de régularité sur lesquelles se repose l’efficacité de PatchMatch pourraient être différentes pour l’axe temporel. Le Tableau 1 montre une comparaison des différents ordres de parcours. Nous évaluons ces ordre de parcours au travers de l’erreur moyenne de patch, par composant de patch. Pour les exemples analysés, on voit que l’ordre de parcours a une très faible influence sur les résultats, et que son choix de l’ordre n’a donc pas une grande importance. Il s’agit toutefois de résultats préliminaires qui mériteraient dêtre approfondis. Enfin,



Trames initiales : « museum » (en haut) et « duo » (en bas) | Résultat de [6] | Notre résultat

FIGURE 2: Nous obtenons des résultats similaires à ceux de [6] avec une accélération d'un facteur 10, et sans intervention de l'utilisateur. Les masques d'occlusion sont surlignés en vert. Nos résultats vidéos sont disponibles sur : http ://www.enst.fr/~gousseau/videoinpainting

il convient de remarquer que PatchMatch comporte une part d'aléatoire. Ainsi, bien que l'algorithme soit efficace en pratique, il n'y a aucune garantie que deux réalisations de l'algorithme complet produisent le même résultat final. Nos expériences montrent néanmoins peu de variabilité des résultats entre différentes réalisations de l'algorithme complet.

2.2.2 Reconstruction de la vidéo

L'étape suivante est la reconstruction de la vidéo. Comme évoqué précédemment, l'inconvénient principal de la reconstruction par moyennage de patchs est l'apparition de flou, ce qui a conduit les auteurs de [5] à proposer une reconstruction par *mean shift*. Néanmoins cette étape est relativement coûteuse et repose sur des paramètres sensibles. Nous proposons une reconstruction alternative beaucoup plus simple et qui en pratique donne des résultats semblables. Nous utilisons la moyenne pondérée de l'équation (1) pour toutes les itérations, exceptée pour la dernière itération au niveau le plus fin, à laquelle la vidéo est reconstruite avec le seul pixel central du patch V_i le plus proche de W_p, pour chaque pixel p. Cette solution peut être vue comme une approche de type recuit simulé (sur le paramètre σ de l'équation (1)) simplifiée à l'extrême.

Une autre partie très importante de la reconstruction est l'initialisation de la solution au niveau le plus grossier de la pyramide. Dans notre algorithme, nous proposons une approche en « pelure d'oignon ». La formule de reconstruction 1 est appliquée par couches concentriques de largeur un pixel, jusqu'à ce que l'occultation soit complètement remplie. L'information contenue dans chaque patch sur la frontière de l'occultation étant incomplète, seule la partie non occultée de chaque patch est comparée lors de la recherche de PPVA. Pour la reconstruction, seuls les PPVAs des patchs dont les centres se trouvent en-dehors de la couche courante de l'occultation sont utilisés.

La dernière partie de l'algorithme est l'utilisation d'une pyramide spatio-temporelle. Pour créer cette pyramide, nous ne sous-échantillonnons pas en temps, sauf pour la vidéo appelée « Jumping girl », pour laquelle la durée d'occultation est particulièrement importante. Avant de sous-échantillonner, nous filtrons la vidéo en moyennant des blocs d'image de taille 2x2. Pour passer d'une échelle grossière à une échelle plus fine, il faut choisir une méthode de sur-échantillonnage de la solution. Dans l'algorithme initial de Wexler, cela est effectué en sur-échantillonnant les PPVAs trouvés à l'échelle grossière, et en reconstruisant l'image avec l'information de l'échelle fine. Cela évite d'avoir une solution d'initialisation trop floutée à la nouvelle échelle. Nous gardons ici cette façon de passer d'une échelle grossière à une échelle plus fine.

3 Résultats

Notre but est de développer une technique d'inpainting vidéo générique permettant d'obtenir automatiquement des résultats de qualité en un temps acceptable. Nous analysons donc aussi bien la qualité visuelle que les temps d'exécutions. Nous nous comparons aux résultats de Wexler *et al.* [5] ainsi qu'à ceux de Granados *et al.* [6]. Notre algorithme est comparé à

Ordre de propagation	Erreur moyenne de PPVA, par composant de patch						
	t,y,x	y,t,x	t,x,y	x,t,y	y,x,t	x,y,t	Recherche exhaustive
Beach Umbrella	9.22	9.61	9.49	9.54	9.57	9.11	6.83
Crossing Ladies	7.53	7.44	7.42	7.51	7.50	7.35	6.14
Jumping Girl	6.48	6.52	6.40	6.49	6.50	6.45	4.80

TABLE 1: Comparaison de l'ordre de propagation, en termes de l'erreur moyenne par composant de patch de tous les plus proches voisins approximatifs trouvés. La dernière colonne représente l'erreur moyenne des plus proches voisins exacts. Les erreurs ont été calculées pour les exemples standards de Wexler *et al*.

Algorithme	Temps de recherche de plus proches voisins approchés				
	Beach Umbrella 264x68x200	Crossing Ladies 170x80x87	Jumping Girl 300x100x239	Duo 960x704x154	Museum 1120x754x200
Wexler [5]	985 s	942 s	7877 s	-	-
Proposé	50 s	28 s	155 s	2515 s	3958 s
Algorithme	Temps d'exécution total				
Granados [6]	11 heures	-	-	-	90 heures
Proposé	21 min	6 min	62 min	7h6min	8h38min

TABLE 2: Temps d'exécution partiels et totaux sur différents exemples. A noter que pour l'exemple « museum », l'algorithme de Granados est parallelisé sur les différents objets occultés et l'arrière plan, alors que le notre ne l'est pas.

celui de [5] en termes de temps de recherche de PPVAs, ce qui constitue l'essentiel du temps d'exécution. Nous présentons aussi nos temps d'exécution totaux. La comparaison des vitesses avec l'algorithme de [6] est plus difficile, ce dernier étant semi-automatique. Cependant, certains temps d'exécution sont donnés dans [6], et nous utilisons ceux-ci comme référence.

On peut voir dans le Tableau 2 les temps de recherche de PPVAs. Nous accélérons cette recherche par un facteur compris entre 20 et 50 par rapport à la méthode de [7]. Pour l'exemple de plus grande taille (« Museum »), notre méthode a nécessité 8h38mins de calcul alors que Granados *et al*. rapportent un temps de 90 heures, sur une architecture d'ordinateur similaire (nous avons utilisé une machine 64-bit avec processeur Intel Xeon à 2.67 GHz). Un seul cœur de la machine a été utilisé dans nos expériences. En revanche, Granados *et al*. parallélisent le traitement des objets occultés et de l'arrière plan, alors que nous traitons ceux-ci simultanément. Un exemple de résultats sur une image de la vidéo Duo est présenté dans la Figure 2. Les résultats sont visuellement très semblables. Les résultats vidéo et les comparaisons avec d'autres méthodes sont disponibles sur : http ://www.enst.fr/~gousseau/videoinpainting.

Remerciements : Nous tenons à remercier Miguel Granados pour avoir bien voulu répondre à nos questions sur son travail.

4 Conclusion

Nous avons montré que de l'inpainting vidéo automatique et générique basé patchs peut être réalisé en un temps raisonnable, même avec des vidéos haute résolution. En généralisant l'algorithme « PatchMatch » au cas spatio-temporel, nous avons en effet obtenu des accélérations d'un facteur 10 au moins, tout en produisant des résultats visuellement très semblables à ceux de [5] et [6], et ce de manière automatique.

Références

[1] S. Masnou and J.-M. Morel, "Level lines based dissoclusion," *ICIP*, vol. 3, pp. 259–263, 1998.

[2] P. Arias, G. Facciolo, V. Caselles, and G. Sapiro, "A variational framework for exemplar-based image inpainting," *Int. J. Computer Vision*, vol. 93, no. 3, pp. 319–347, 2011.

[3] K. A. Patwardhan, G. Sapiro, and M. Bertalmio, "Video inpainting of occluding and occluded objects," *ICIP*, vol. 2, pp. 69–72, 2005.

[4] M. V. Venkatesh, S-C. S. Cheung, and J. Zhao, "Effcient object based video inpainting," *ICIP*, vol. 30, pp. 168–179, 2006.

[5] Y. Wexler, E. Schechtman, and M. Irani, "Space-time completion of video," *IEEE Trans. PAMI*, vol. 29, no. 3, pp. 463–476, 2007.

[6] M. Granados, J. Tompkin, K. Kim, O. Grau, J. Kautz, and C. Theobalt, "How not to be seen – object removal from videos of crowded scenes," *Eurographics*, vol. 31, pp. 219–228, 2012.

[7] S. Arya and D. Mount, "Approximate nearest neighbor queries in fixed dimensions," *SIAM*, 1993.

[8] C. Barnes, E. Schechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch : A randomized correspondence algorithm for structural image editing," *SIGGRAPH*, vol. 28, no. 3, 2009.

[9] S. Darabi, E. Schechtman, and C. Barnes, "Image melding : Combining inconsistent images using patch-based synthesis," *SIGGRAPH*, vol. 31, 2012.