Université Claude Bernard
Lyon I

Licence d'informatique

Effacement d'un Objet d'une Photo Par Synthèse Basée Exemplaire

14 Juin 2004
Travail d'Études et de Recherches

Jean-Marc TRÉMEAUX
Encadrant : Jean-Claude IEHL

Université
Claude Bernard Lyon1 UFR d'informatique

Table des matières

Résumé

Dans ce document, nous allons présenter une méthode pour supprimer un objet dans une image, et reconstituer la partie supprimée. Ce travail, habituellement réalisé par un graphiste avec un outil de retouche d'images, est ici effectué automatiquement et en un laps de temps très court (de l'ordre de quelques secondes). Le but de la reconstitution est de rendre une image visuellement plausible. La solution présentée ici est une méthode itérative, qui consiste à diviser la région manquante de l'image en petits fragments rectangulaires, puis substituer à chaque fragment une partie de l'image connue de façon à ce que le résultat soit cohérent. Nous présenterons la méthode employée, ainsi qu'une implémentation. Enfin, nous fournirons divers exemples à partir desquels nous vérifierons l'efficacité de cette méthode.

Mots clés : suppression d'objet, synthèse de texture, fragment, synthèse basée exemplaire, composition

FIG. 1 – Fenêtre principale de l'application, détourage de l'objet à supprimer

1 Introduction

Lors de le retouche de photographies ou d'images, le graphiste est souvent amené à supprimer des portions d'image. C'est un travail fastidieux qui est accompli avec des outils de sélection, copie, collage et "clone" d'image. Le graphiste séléctionne manuellement une partie de l'image à recopier, et duplique cette image sur une zone de l'objet à effacer.

Un certain nombre d'algorithmes ont été proposé pour effectuer cette tache automatiquement. On peut identifier deux classes de méthodes. La première technique, dite *basée exemplaire*, consiste à génerer une nouvelle texture à l'interieur de la région à effacer, en copiant des fragments d'une image utilisée comme source. Cette méthode [2, 3, 4] est simple et peu couteuse, car l'image est synthétisée par blocs entiers. La seconde technique, dite d'*inpainting*, vise à diffuser l'information du bord à l'intérieur de la zone à effacer. Cette méthode permet de propager les structures linéaires, ou *isophotes* dans la région cible.

Le travail ici présenté est une implémentation de la méthode proposée dans le document [1]. Il s'agit à la base d'une synthèse de texture basée exemplaire, elle en hérite donc de son efficacité. L'algorithme peut cependant être amélioré pour prendre en compte la propagation des structures, via un bon ordonnancement des blocs à remplacer.

1.1 Donnée du problème

Le problème peut être posé de la façon suivante :

Données : Image I, chemin non nul $\delta\Omega$ défini par l'utilisateur, partitionnant l'image I en deux régions : Φ (région source) et Ω (région cible), et taille d'un exemplaire (hauteur, largeur).

Résultat : Image I complétée, la région cible Ω ayant été remplie avec des exemplaires de la région source Φ.

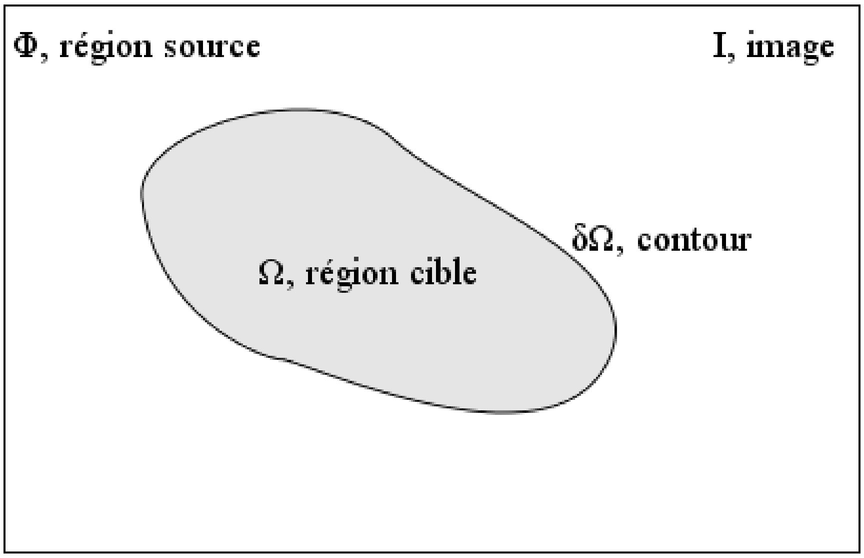

FIG. 2 – Donnée du problème

2 Synthèse basée exemplaire

2.1 Algorithme général

La première étape consiste à spécifier le contour $\delta\Omega$. Ce contour est saisi par l'utilisateur, via un outil de détourage. À chaque pixel p est associée une valeur de *confiance* C(p), 1 dans la région source Φ et 0 dans la région cible Ω. L'utilité de cette valeur de confiance sera précisée plus tard.

Nous identifions ensuite tous les blocs qui serviront d'*exemplaires*. Cet ensemble d'exemplaires est défini à l'initialisation et ne changera pas par la suite. La région source est définie comme l'image totale, privée la région à supprimer. On partitionne la région source Φ en blocs adjacents Ψ_p, de taille identique et fournie par l'utilisateur. Ces blocs doivent être compris entierement dans la région source Φ. Les blocs chevauchant le contour $\delta\Omega$ sont rejetés.

La boucle principale consiste à faire diminuer la surface de la région cible Ω jusqu'à ce que celle-ci soit nulle. Tout d'abord, on identifie un ensemble de blocs Ψ_q sur le contour $\delta\Omega$. On affecte à chacun de ces blocs une priorité, qui est fonction de la quantité d'information dont on dispose sur cette portion de l'image.

L'étape suivante consiste à identifier le bloc du contour qui à la plus haute priorité. Ce bloc Ψ_q est comparé à tous les exemplaires Ψ_p : on calcule la distance $d(\Psi_p, \Psi_q)$ qui est la somme des distances euclidiennes, sur l'espace RVB, des pixels correspondants de Ψ_p et Ψ_q.

L'exemplaire Ψ_p le plus "proche" de Ψ_q est alors recopié tel quel sur Ψ_q.

La derniere étape consiste à mettre à jour les indices de confiance P(p) pour chaque pixel $p \in \Psi_q \cap \Omega$, puis à calculer le nouveau contour $\delta\Omega^{t+1}$.

2.2 Résumé de l'algorithme

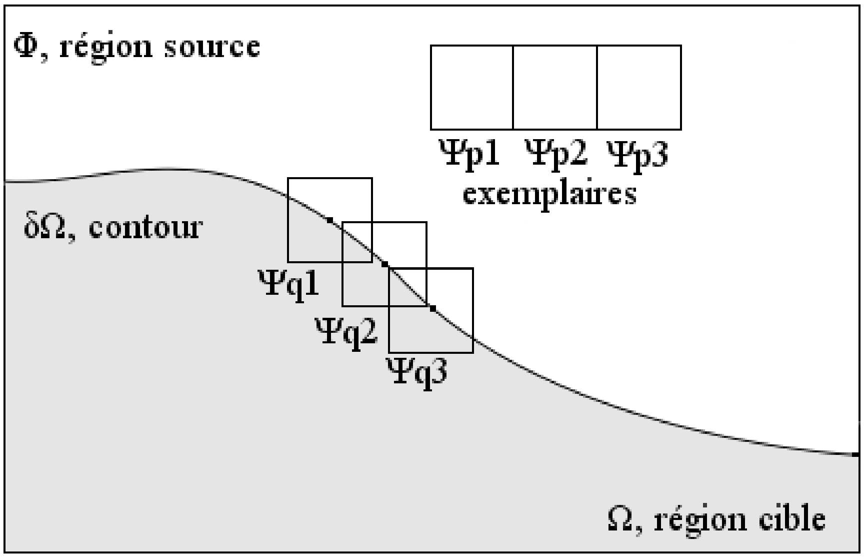

FIG. 3 – Sélection des exemplaires

1. Identifier le contour $\delta\Omega^0$.
2. Identifier un ensemble d'exemplaires Ψ_p inclus dans Φ.
3. Tant que $\delta\Omega^t \neq \emptyset$, faire :
   3.a Identifier un ensemble de blocs Ψ_q du contour $\delta\Omega^t$
   3.b $\forall q \in \delta\Omega^t$ calculer la priorité P(q)
   3.c Identifier le bloc Ψ_{qmax} qui a la plus haute priorité
   3.d Trouver l'exemplaire Ψ_p qui minimise $d(\Psi_p, \Psi_{qmax})$
   3.e Recopier l'image de l'exemplaire Ψ_p sur le bloc du contour Ψ_{qmax}
   3.f Mettre à jour les valeurs de confiance C(p) pour chaque pixel de $\Psi_{qmax} \cap \Omega$, puis mettre à jour le contour $\delta\Omega$.

Fin tant que

2.3 Ordonnancement des blocs

L'étape 3.c de l'algorithme nécessite d'identifier un bloc du contour afin de le remplir. L'idée est de privilégier les blocs pour lesquels on dispose du maximum d'information.

Pour ce faire, on assigne à chaque pixel p de l'image I une valeur *confiance*. Cette valeur vaut initialement 1 pour un pixel appartenant à la région source Φ, et 0 pour un pixel de la région cible Ω. La valeur est fixée lorsque l'on copie un pixel vers Ω, et elle ne change plus après.

La valeur de confiance d'un bloc q est calculée de la façon suivante : c'est la somme des valeurs de confiance de tous les pixels de ce bloc, divisée par la surface du bloc.

$$C(p) = \frac{\sum_{r \in \Psi_q} C(r)}{|\Psi_q|}$$

où $|\Psi_q|$ représente la surface de Ψ_q.

On note que la *confiance* d'un pixel (ou d'un bloc) est une valeur réelle comprise entre 0 et 1. La valeur 1 dénote une confiance absolue dans la valeur de ce pixel, ce qui est le cas pour la région source Φ définie initialement, alors que la valeur 0 représente un pixel inconnu.

Lorsque l'on remplit un bloc Ψ_q, on fixe la valeur de confiance C(r) des pixels copiés dans $\Psi_q \cap \Omega$: la valeur de confiance du bloc Ψ_q est affectée à tous les pixels recopiés dans ce bloc.

$$C(r) = C(q) \; \forall r \in \Psi_q \cap \Omega$$

Ainsi, les valeurs de confiance se propagent en décroissant vers l'interieur de la région cible. Cela traduit le fait que l'on a un maximum d'information sur la région source Φ, et que l'information va en décroissant lorsque l'on progresse à l'interieur de la région cible Ω.

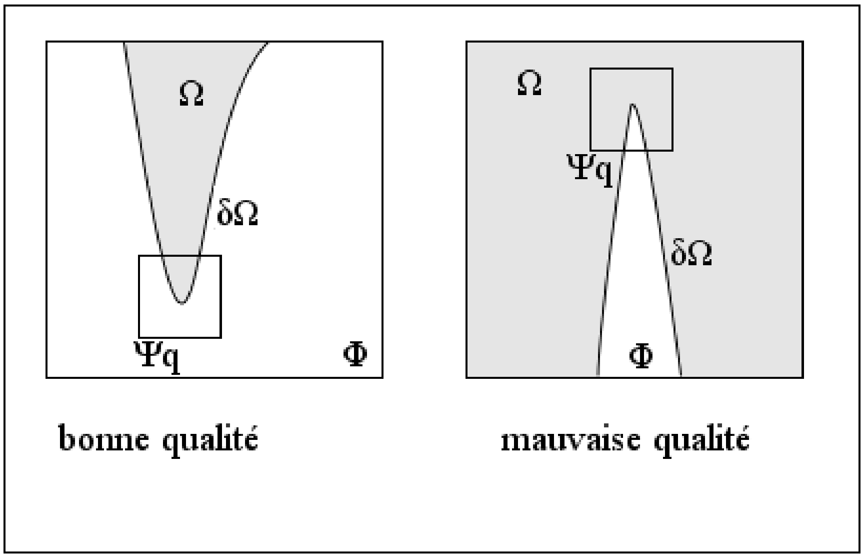

FIG. 4 – Qualité des blocs du contour. Le bloc de gauche contient plus d'information valide sur les pixels environnants que celui de droite.

3 Améliorations proposées

3.1 Calcul de la dissimilarité des blocs

Lors de la comparaison des blocs des blocs du contours aux exemplaires, nous avons calculé la distance euclidienne $d(\Psi_p, \Psi_{qmax})$ sur l'espace RVB, pour des raisons de facilité. En effet, l'image est chargée à partir de cet espace de couleur. Notre objectif étant de rendre la texture synthétisée cohérente avec le reste de l'image, il serait judicieux de calculer la distance des pixels sur l'espace CIE, celui-ci ayant de meilleurs propriétés sur la perception humaine des couleurs [5].

3.2 Prolongement des structures linéaires

Si cette méthode de synthèse basée exemplaire propage correctement la texture, elle est en l'état incapable de détecter et propager les structures linéaires. L'amélioration consiste à changer l'ordonnancement des blocs. La priorité d'un bloc Ψ_q du contour serait définie comme suit :

$$P(q) = C(q)D(q)$$

où C(q) est le terme de *confiance* défini précédemment, et D(q) le terme de *donnée*, compris entre 0 et 1. Cette valeur est d'autant plus élevée que Ψ_q se situe sur la continuité d'une structure linéaire.

4 Implémentation

4.1 Travail effectué

L'implémentation de cet algorithme s'est fait en deux temps. Tout d'abord, un programme en ligne de commande a été réalisé. Ce programme prend en paramètres une image d'entrée au format "brut" RVBA, ainsi qu'un masque préalablement créé sous un logiciel de retourne d'image, définissant les régions source et cible. Il renvoie l'image complétée dans un fichier d'image au format brut.

La deuxième partie est la création d'une interface grâce à l'outil GTK+ [6]. Cette interface permet d'ouvrir une image dans un format communément utilisé (PNG, JPEG...), de définir manuellement le contour $\delta\Omega^0$ et enfin d'effectuer le calcul et d'afficher le resultat.

Ce programme a été testé sur la machine de développement (Intel Pentium 700MHz) et effectue des calculs de l'ordre de quelques secondes (pour des exemplaires de taille 9x9) ou quasi-instantanément pour des exemplaires de taille supérieure.

4.2 Mode d'emploi

Le programme s'utilise de la façon suivante :

- Ouvrir une image avec le menu Fichier->Ouvrir.
- Détourer l'objet en effectuant des clics de souris. En cas d'erreur, la sélection peut être annulée par le menu Suppression->Annuler la sélection.
- Lancer le processus de suppression par le menu Suppression->Supprimer.

5 Exemples

Voici pour finir une série d'exemples de suppression d'objet sur des photographies. Les exemples 1 à 4 montrent l'efficacité de l'algorithme lorsque l'objet se situe sur un fond uniforme.

L'exemple 5 exhibe le problème lié à la propagation des structures linéaires. La texture du mur est bien propagée, cependant il apparait clairement que les blocs n'ont pas été remplacés dans le bon ordre. La deuxième amélioration proposée permettrait de corriger ce défaut.

FIG. 5 – Exemple 1

FIG. 6 – Exemple 2

FIG. 7 – Exemple 3

FIG. 8 – Exemple 4

FIG. 9 – Exemple 5

6 Conclusion

Nous avons développé une méthode pour effacer un objet d'une image. Si nous devons retenir un point positif pour cette application, c'est sa rapidité. Un rendu prend en effet un temps de l'ordre de quelques secondes seulement, contre plusieurs heures pour d'autres techniques. Le rendu pourrait cependant à être amélioré. Les structures complexes ne sont pas bien reproduites, et un résultat correct est obtenu uniquement avec un choix adéquat pour la taille des exemplaires. Les améliorations à apporter à cet algorithme porteraient principalement sur un meilleur ordonnancement des blocs du contour.

Références

[1] Antonio Criminisi, Patrick Perez, Kentaro Toyama Region Filling and Object Removal by Exemplar-Based Image Inpainting (2003). *http ://research.microsoft.com/research/pubs/view.aspx ?tr_id=694*

[2] M. Ashikhmin Synthesizing natural textures (Mars 2001). *Proc. ACM Symposium on Interactive 3D Graphics*, pages 217-226, Research Triangle Park, NC

[3] A. Efros, W.T. Freeman Image quilting for texture synthesis and transfer (Août 2001). *Proc. ACM Conf. Comp. Graphics (SIGGRAPH)*, pages 341-346, Eugene Fiume

[4] A. Efros, T.Leung Texture synthesis by non-parametric sampling (Sept. 1999). *Proc. Int. Conf. Computer Vision*, pages 1033-1038, Kerkyra, Greece

[5] J.M. Kasson, W. Plouffe An Analysis of selected computer interchange color spaces (Oct. 1992). *ACM Transactions on Graphics*, volume 11, pages 373-405

[6] GTK+ The GIMP Toolkit *http ://www.gtk.org/*