Rapport Projet Final

Traitement d'images - IFT 6150

Max Mignotte

Par

Achille Sowa

Université de Montréal

Introduction

L'article sur lequel est basé notre projet est un article publié en 2006 par Jin Wang, Yanwen Guo, Yiting Ying, Yanli Liu et Qunsheng Peng. Il est intitulé **Fast Non-local algorithm for image denoising** et décrit une méthode rapide d'un algorithme de débruitage non local et non linéaire. Dans un premier temps, les auteurs présentent une version non optimisée de l'algorithme dont la complexité théorique est de $\Theta(n^2)$ pour une image de taille bruitée de taille nxn. Ensuite, ils décrivent une méthode pour réduire le temps d'exécution de l'algorithme présenté d'un facteur de **50**. Si la complexe théorique de l'algorithme reste inchangée, cette réduction rend l'algorithme utilisable en pratique. Par exemple, une image de 500MB peut être traitée en moins de 10 secondes avec la version optimisée, ce qui prendrait plus de 5 minutes avec la version non optimisée.
Dans ce rapport, nous allons présenter en détail la version non optimisée et la version optimisée de l'algorithme telle que décrite dans l'article original. Ensuite nous présenterons notre implémentation de la version non optimisée et quelques résultats obtenus. Nous mettrons ces résultats en relation avec ceux obtenus par l'algorithme de débruitage par ondelettes de Haar que nous avons implémenté lors du TP3.

Algorithme non optimisé

Soit une image $I = \{I(x, y)|(x, y) \in \Omega\}$ bruitée. L'objectif de l'algorithme est de produire une image $I' = \{I'(x, y)|(x, y) \in \Omega\}$ débruitée. Pour déterminer la valeur finale de chaque pixel $I'(xi, yi)$ dans l'image finale détruitée, l'algorithme calcule pour ce pixel, la similarité entre son voisinage de taille MxM et celui des n^2-1 autres pixels de l'image. L'idée ensuite est de faire une moyenne de tous les points de l'image, en donnant un poids plus élevé aux pixels dont le voisinage est similaire au voisinage du pixel considéré, et un poids plus faible à ceux dont le voisinage est différent du voisinage du pixel considéré.
Soit N_i et N_j correspondants aux voisinages des pixels de (x_i, y_i) et (x_j, y_j). La distance entre les deux voisinages est évaluée par la norme euclidienne

$$S(i, j) = ||N_i - N_j||^2$$

Une fois obtenu $S(i,j)$ pour chaque couple de pixel, le poids du pixel (x_j, y_j) dans le moyennage final pour l'obtention de $I'(x_i, y_i)$ est déterminé par

$$w(i,j) = \frac{1}{Z(i)} e^{-\frac{S(i,j)}{h^2}}$$

Avec Z(i) $= \sum_{j=1}^{n^2} e^{-\frac{S(i,j)}{h^2}}$ la constante de normalisation et h une constante proportionnelle à la variance du bruit présent dans l'image. Nous remarquons que pour un i fixé, pour tout j, $w(i,j) > 0$ et $\sum_j w(i,j) = 1$, donc on a en quelque sorte une distribution de probabilité sur l'ensemble des pixels de l'image où les pixels les plus semblables au pixel considéré, où qu'ils se trouvent dans l'image ont un grand poids dans l'évaluation de sa valeur finale.

Pour calculer chaque $S(i,j)$, on a besoin de M^2 opérations et comme il faut le faire pour tout couple de pixels, cela fait en moyenne $M^2 n^4$ opérations. Cependant, en pratique on utilise $M = 7$ et la similarité n'est pas recherchée tout au long de l'image mais seulement dans une fenêtre de taille $21x21$ autour du pixel considéré. On a donc dans l'ordre de $49 * 441 * n^2$ opérations pour le remplissage de la matrice $S(i,j)$. Une fois la matrice $S(i,j)$ obtenu, la matrice $w(i,j)$ s'obtient par n^2 estimations de $Z(i)$ chacune nécessitant en moyenne 441 opérations et donc $w(i,j)$ s'obtient avec environ $2 * 441 * n^2$ operations. Pour chaque $I'(x_i, y_i)$ on fait un moyennage avec $w(i,j)$ de taille $441*n^2$ et donc $441*2*n^2$ operations. On a donc environ

$$2 * 441 * n^2 + 2 * 441 * n^2 + 49 * 441 * n^2$$

On voit aisement que le facteur le plus important dans cette sommation est le terme $49 * 441 * n^2$ qui correspond au remplissage de la matrice $S(i,j)$. La version améliorée de l'algorithme consiste en une optimisation de ce facteur.

Algorithme optimisé

Premièrement, les auteurs remarquent que:

$$S(i,j) = \|N_i - N_j\|^2 = \sum_{l=0}^{M-1}\sum_{m=0}^{M-1}[I_i(l,m) - I_j(l,m)]^2$$

Avec $I_i(l,m)$ et $I_j(l,m)$ correspondent au pixels des voisinages N_i et N_j. Avec $I_i(l,m)$ et $I_j(l,m)$ correspondent au pixels des voisinages N_i et N_j. En représentant $I_j(l,m)$ par ses coordonnées dans l'image miroir (figure 1), on ne change pas la valeur du voisinage, mais on a maintenant $I_j(l - x'_j, m - y'_j)$ avec $x'_j = 3M/2 + x_j$ et $y'_j = 3M/2 + y_j$ et on obtient

$$\begin{aligned}
S(i,j) &= \sum_{l=0}^{M-1}\sum_{m=0}^{M-1}[I_i(l,m) - I_j(l - x'_j, m - y'_j)]^2 \\
&= \sum_{l=0}^{M-1}\sum_{m=0}^{M-1} I_i(l,m)^2 + I_j(l - x'_j, m - y'_j)^2 - 2I_i(l,m)I_j(l - x'_j, m - y'_j) \\
&= \sum_{l=0}^{M-1}\sum_{m=0}^{M-1} I_i(l,m)^2 + \sum_{m=0}^{M-1} I_j(l - x'_j, m - y'_j)^2 - 2\sum_{m=0}^{M-1} I_i(l,m)I_j(l - x'_j, m - y'_j) \\
&= \sum_{l=0}^{M-1}\sum_{m=0}^{M-1} I_i(l,m)^2 + \sum_{m=0}^{M-1} I_j(l,m)^2 - 2\sum_{m=0}^{M-1} I_i(l,m)I_j(l - x'_j, m - y'_j) \\
&= N_i^2 + N_j^2 - 2N_i * N_j
\end{aligned}$$

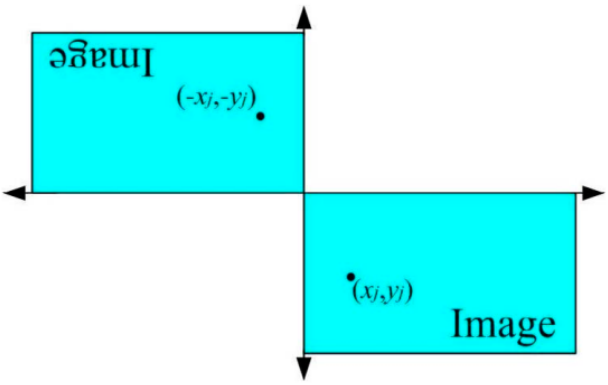

Figure 1: Image mirroir

On remarque la convolution $N_i * N_j$. Pour $N \geq 7$, elle est plus efficace dans le domaine de spectral avec la FFT, ce qui nous permet d'efficacement évaluer le terme $2N_i * N_j$ Pour ce qui est de N_i, dans un premier temps on évalue la matrice intégrale de l'image SSI qui est définie par

$$SSI(x_0, y_0) = \sum_{x_0, y_0 I(x,y)^2}$$

On peut l'évaluer efficacement à l'aide du code de programmation dynamique en remplissant la machine ligne par ligne comme dans l'algorithme 1

Algorithm 1: Algorithme pour déterminer la matrice intégrale de l'image I

Input: image I de taille nxn

Output: matrice integrale SSI de taille nxn

SSI(0,0) = I(0,0)^2**for** $i = 1$ **to** $n - 1$ **do**
| SSI(i,0) = SSI(i-1, 0) + I(i,0)^2

for $j = 1$ **to** $n - 1$ **do**
| SSI(0,j) = SSI(0,j-1) + I(0,j)^2

for $i = 1$ **to** $n - 1$ **do**
| **for** $j = 1$ **to** $n - 1$ **do**
| | SSI(i,j) = SSI(i-1,j) + SSI(i,j-1) - SSI(i-1,j-1) + I(i,j)^2

return SSI

Une fois SSI préremplie, et à l'aide de l'image de la figure 2, on évalue

$$S_D = S_{A\cup B\cup C\cup D} + S_A - S_{A\cup B} - S_{A\cup C} = SSI(x_1, y_1) + SSI(x_0, y_0) - SSI(x_0, y_1) - SSI(x_1, y_0)$$

Ainsi on obtient N_i et N_j en temps constant (une fois la matrice SSI remplie)

Et donc le calcul de S(i,j) demande environ $441 * n^2 + n^2$ opérations, ce qui est bien moindre que les $49 * 441 * n^2$ opérations de départ. PS Nous ne comprenons pas exactement comment les auteurs du papier reduisent à un temps constant l'évaluation de tous les $N_i * N_j$

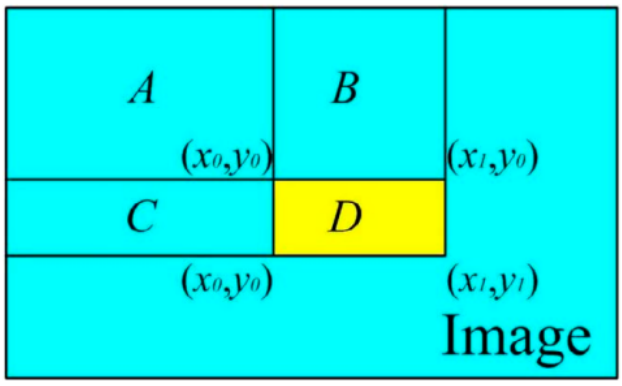

Figure 2: Matrice integrale

Implémentation et résultats

Nous avons implémenté la version non optimisée de l'algorithme proposé dans l'article. Afin que l'expérimentation fonctionne correctement, nous avons dû prendre en compte quelques points importants:

1. D'abord dans la gestion des erreurs numériques. Le calcul de $w(i,j)$ fait intervenir une exponentielle de valeurs négatives très grandes en valeurs absolues. Si on ne fait pas attention, on peut se retrouver avec des poids qui sont soit tous nuls. Pour pallier ce problème, on a commencé par réduire de leur valeur maximale toutes les valeurs en exposant. Ceci a pour effet de produire un algorithme plus stable numériquement. Cette astuce est implémentée dans notre fonction `softmax`

2. Nous avons gardé $S(i,j)$ dans une matrice de n^2 lignes et M^2 colonnes. Chaque ligne i de cette matrice contient ainsi les coefficients(i,j) correspondants. Il fallait alors faire effectuer un changement de repère pour passer de la case (i,j) de la matrice S aux valeurs des pixels correspondants. Cela a été implémenté dans les fonctions `similarity` et `weights` de notre code source.

La figure 4 représente le résultat de l'algorithme sur l'image **photograph.pgm**, bruitée avec un bruit gaussien de variance $\sigma^2 = 400$. Nous avons évalué l'ISNR de cette image et il est de 5.79, supérieur à celui obtenu pour un algorithme de débruitage par les ondelettes de Haar vu en TP3 qui était de 4.65.

Figure 3: Image bruitee

Figure 4: Image debruitee

Figure 5: Image originale