

Rapport d'introduction à la recherche

Traitement de l'image



Samuel BURY Loqman KHALFAOUI

16/11/2022

E4FI

Synthèse de l'article

1. Introduction

Cette article intitulé “EXTREME IMAGE COMPLETION” écrit par Radhakrishna Achanta, Nikolaos Arvanitopoulos et Sabine Susstrunk dans le cadre de l'école “School of Computer and Communication Sciences Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland” traite du sujet de la complétion d'image.

La problématique traité par l'article est la reconstitution d'une image à partir d'une image dont 99% des pixels sont aléatoirement manquant.

Le but de l'article est de répondre à la problématique de complétion d'image en utilisant un algorithme de complexité linéaire liée au nombre de pixel total de l'image, et en temps réel. L'algorithme est donc plus efficace et plus rapide que les méthodes existantes citées dans l'article. La méthode présentée est donc particulièrement pertinente.

Pour valider l'hypothèse, le test, on compare la méthode présentée EFAN avec les méthodes TV, PDE , FA, selon les critères MSE, PSNR et SSIM.

Nous allons à présent expliquer ces critères, ainsi que comment les interpréter

L'indice SSIM:

$$SSIM(x, y) = l(x, y) \cdot c(x, y) \cdot s(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_x\sigma_y + c_2)(cov_{xy} + c_3)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)(\sigma_x\sigma_y + c_3)}$$

Cette indice compris entre 0 et 1 permet de mesurer la similitude de la structure d'une image avec avec l'image original, la perception des structures est plus accessible à l'humain que le comparatif pixel à pixel, plus le SSIM se rapproche de 1, plus la structure de l'image se rapproche de l'image original. Comme on peut le voir dans le comparatif de l'article entre les 3 méthodes, comme on peut le voir

sur la figure 3 de l'article, le SSIM de la technique EFAN est en général nettement supérieur à la technique TV, et légèrement supérieur à la technique PDE et FAN.

L'indice MSE :

L'indice MSE (Mean Squared Error, ou erreur quadratique moyenne) est un indicateur de la qualité de la différence entre deux images numériques ou deux séquences vidéo. Il mesure la moyenne quadratique des différences entre chaque pixel de l'image originale et chaque pixel de l'image comparée. Plus précisément, l'indice MSE est défini comme la somme des différences au carré entre chaque pixel de l'image originale et chaque pixel de l'image comparée, divisée par le nombre total de pixels dans l'image.

Plus la valeur de l'indice MSE est faible, meilleure est la qualité de l'image comparée par rapport à l'image originale.

Ici, le MSE est plus petit, en général que les autres techniques, cependant, dans un cas, la méthode PDE possède un MSE plus petit.

L'indice PSNR (db) :

Le PSNR (Peak Signal-to-Noise) est un indice de qualité de l'image utilisé pour mesurer la différence entre une image originale et une image altérée ou compressée. Plus le PSNR est élevé, moins l'image altérée ou compressée est différente de l'image originale, et donc plus la qualité de l'image est considérée comme bonne.

Le PSNR est calculé en utilisant la formule suivante :

$$\text{PSNR} = 10 * \log_{10}(\text{MAX}^2 / \text{MSE})$$

où MAX est la valeur maximale possible d'un pixel dans l'image (par exemple, 255 pour une image en 8 bits par pixel) et MSE (Mean Squared Error) décrit ultérieurement.

Selon le comparatif, la technique EFAN est globalement supérieure aux autres méthodes sauf pour le PDE qui possède un PSNR supérieur

Cependant, il convient de noter que le PSNR ne tient pas compte de tous les aspects de la qualité perçue par l'œil humain, et qu'il peut donc ne pas refléter de manière précise la qualité perçue par un humain.

Concernant les résultats du textes, basé sur les textes de l'article, nous constatons, que la méthode EFAN donne un rendu supérieur qualitativement que les autres méthodes cependant, la méthode PDE peut avoir une meilleure qualité dans un certain cas qu'il serait intéressant de déterminer.




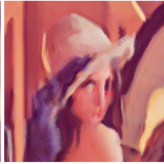



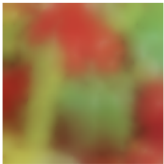





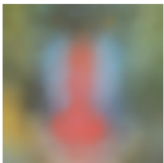

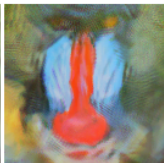


(a) Original	(b) TV	(c) PDE	(d) Exemplar	(e) FAN	(f) EFAN
					
MSE	1288.32	481.80	542.48	503.55	461.55
PSNR (dB)	17.03	21.30	20.79	21.11	21.49
SSIM	0.86	0.92	0.88	0.91	0.92
					
MSE	1694.05	665.54	1098.15	679.71	620.92
PSNR (dB)	15.84	19.90	17.72	19.81	20.20
SSIM	0.77	0.87	0.80	0.88	0.88
					
MSE	1692.16	1063.90	1595.77	1140.08	1127.57
PSNR (dB)	15.85	17.86	16.10	17.56	17.61
SSIM	0.38	0.52	0.44	0.51	0.52
Avg. speed (s)	1200+	2.64	7200+	0.07	0.03

Fig. 2. Visual comparison of extreme image completion - images are recreated using only 1% randomly selected original pixels (using the same seed for random number generation). In terms of signal reproduction quality and computational efficiency, the proposed methods of FAN and EFAN (columns e and f) outperform the state-of-the-art. All images are of size 512×512 . Images are best viewed on display screen by zooming-in.

Nous trouvons cet article intéressant, dans la mesure où ces différentes méthodes nous ont impressionné. Il était compliqué pour nous de comprendre comment reconstruire une image à partir d'1% des items tellement cela nous semblait surréaliste.

De plus, la méthode EFAN présentée est d'une complexité linéaire au nombre de pixel, contrairement aux autres qui sont de complexité géométrique, ce qui permet de faire du traitement en temps réel.

Concernant l'article, il est très bien écrit et concis, L'algorithme EFAN est assez bien expliqué.

Etude de l'implémentation

Le code fournis en C++ et en langage Matlab concerne la méthode FAN et EFAN.

Il nous a donc été possible de comparer le traitement de ces 2 techniques, cependant, aucun algorithme concernant le calcul des indices n'a été fourni, nous comparons donc les résultats à l'œil nul.

Le code est bien structuré, il consiste en une fonction en C++,

mexFunction () :

La fonction mexFunction prend plusieurs arguments en entrée et en sortie :

- nlhs (nombre de sorties du côté gauche) est le nombre de variables de sortie que le programme doit renvoyer à MATLAB.
- plhs (pointeurs de sorties du côté gauche) est un tableau de pointeurs vers les variables de sortie qui seront renvoyées à MATLAB.
- nrhs (nombre d'arguments du côté droit) est le nombre d'arguments en entrée que le programme a reçus de MATLAB.
- prhs (pointeurs d'arguments du côté droit) est un tableau de pointeurs vers les arguments en entrée reçus de MATLAB.

La fonction mexFunction peut être utilisée pour effectuer des opérations de traitement de données sur les arguments en entrée, puis renvoyer les résultats sous forme de variables de sortie à MATLAB. Dans le cas présent, il semble que la fonction effectue du

traitement de l'image, tels que la conversion de l'image en niveaux de gris ou la conversion de l'espace de couleur de l'image de RGB à LAB, et renvoie l'image traitée et un masque comme sorties.

Cette fonction n'est utilisée que dans les programmes compilés avec mex. d'où les commentaires donnant des instructions pour la compilation.

Dans le fichier Demo.M , on va mettre notre algorithme en situation en **Le code** commence par charger une image depuis un fichier et en définir un pourcentage de pixels à sauvegarder (c'est-à-dire remplir). Il génère ensuite un masque de pixels à sauvegarder en sélectionnant aléatoirement un certain nombre de pixels dans l'image et en les marquant dans le masque.

Le code appelle ensuite la fonction `efan_mex` avec l'image et le masque en entrée, et stocke le résultat dans la variable `outimg`. La fonction `efan_mex` effectue le traitement de l'image et renvoie l'image complétée.

Le code affiche finalement l'image complétée en utilisant la fonction `imshow`.

Concernant la qualité du code , il est relativement bien organisé et indenté, ce qui le rend facile à lire et à suivre. Les variables ont des noms clairs et descriptifs, ce qui contribue également à la lisibilité du code. Cependant, il y a un nombre important de commentaires et de lignes vides qui peuvent rendre le code un peu moins facile à parcourir.

De plus, le code est bien documenté, ce qui peut faciliter sa compréhension pour les personnes qui souhaitent le modifier. Cependant, il y a de nombreuses directives `#include` et une fonction `mexFunction` qui sont spécifiques à la compilation de code en utilisant la commande `mex`, ce qui va rendre le code plus facile à comprendre pour les personnes qui ne sont pas familiarisées avec cette commande.

Le code correspond exactement à ce qui est indiqué dans l'article. Le code est même plus compréhensible.

Expérimentations

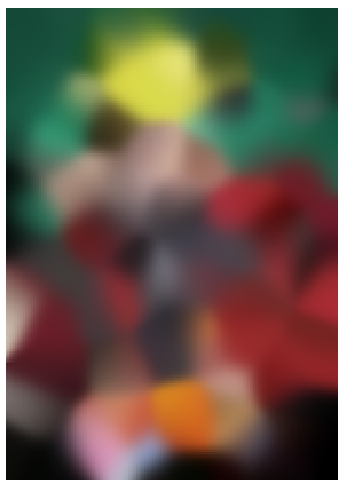
Voici ce que nous avons décidé d'expérimenter avec cette algorithmme:

- Reproduire la photos de la femme au chateau dans l'exemple,
- reproduire une photos prise avec un Iphone
- reproduire un dessin manuel
- reproduire un dessin d'infographie

Dans tous ces cas de figure nous avons gardé le nombre de pixel restant de chaque image à 1%.

Voici dans un tableau le résumé des résultats

Original	Reproduite
	



Conclusion de l'expérimentation, la méthode étant sensible aux intensité maximum des pixels, la méthode fonctionne beaucoup mieux sur une photo de bonne qualité que sur des dessins infographies, présentant une moyenne d'intensité des pixels supérieurs.

Merci