Projet IMA 201 - Augmentation de la profondeur de champs par fusion d'image

Raphael ACHDDOU, Matthis Maillard

novembre 2017

Première partie

La fusion d'image basée sur le filtrage guidé

Les photographies sont de plus en plus importantes dans nos vies. Aujourd'hui des millions de personnes s'improvisent photographes grace à leurs smartphones qui font pour eux des photos nettes bien exposées. Les méthodes présentées dans l'article proposé vont dans ce sens. En effet cet article explique comment faire différentes photographies d'un ou plusieurs objets avec des expositions différentes et des mises au point différentes, comment obtenir une image la plus nette possible et la mieux exposée possible. Dans un premier temps, nous expliquerons comment cette méthode marche, puis dans un second temps nous montrerons nos tests et les critiques que l'on a pu en tirer.

1 Décomposition à deux niveaux des images de bases

Appelons les images d'entrées les I_n. On commence par obtenir les calques de base $B_n = I_n * Z$ où Z est un filtre moyen de taille $31x31$. On obtient ensuite les calques de détails Dn construits par différence : $D_n = I_n - B_n$

(a) $.I_1$ (b) $.B_1$ (c) $.D_1$

FIGURE 1 – les différents calques pour l'image 1

(a) $.I_2$ (b) $.B_2$ (c) $.D_2$

FIGURE 2 – les différents calques pour l'image 2

Ainsi, nous avons dans un calque une représentation des variations à grande échelle et dans l'autre une représentation des variations à petite échelle.

2 Construction de cartes de poids intermédiaires P_n

On commence par chercher les fortes variations avec un filtre Laplacien, que l'on applique à chaque image en entrée In. On obtient $H_n = I_n * L$ avec L le filtre Laplacien de taille 3x3 :

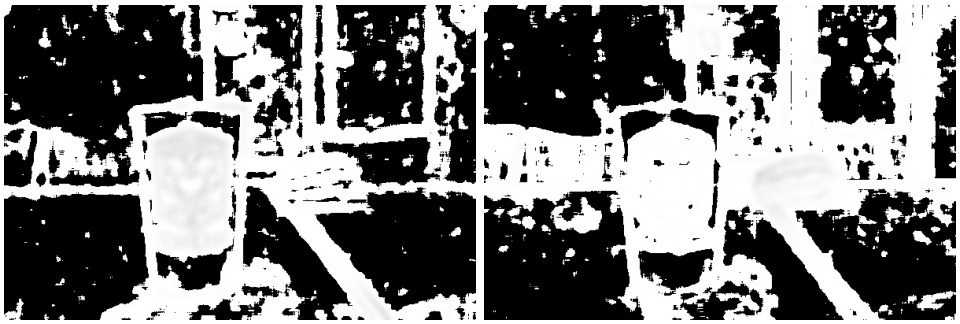
$$L = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Le résultat est souvent très bruité, c'est pourquoi on convole la valeur absolue de cette matrice (qui

peut avoir des valeurs négatives) avec un filtre gaussien qui lisse l'image. On obtient une carte de saillance S_n qui représente assez bien les détails de chaque image d'entrée :

$$S_n = |H_n| * g_{r_g, s_g}$$

avec g le filtre gaussien et r_g, s_g fixé à 5. On obtient les images de saillances suivantes :



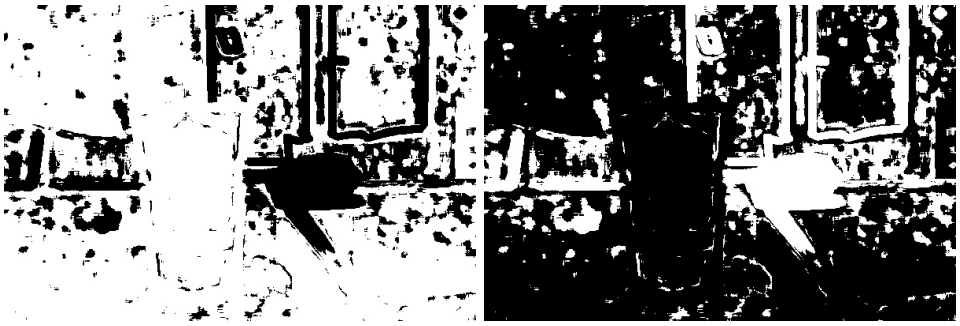
(a) S_1 (b) $.S_2$

FIGURE 3 – les images de saillances pour I_1 et I_2

On souhaite ensuite construire des cartes de poids pour chaque image qui donneraient l'importance de chaque image dans l'image finale en quelque sorte. On construit donc des cartes de poids P_n :

$$P_n^k = \begin{cases} 1 \; si \; S_n^k & = & max(S_1^k, ..., S_N^k) \\ 0 \; sinon \end{cases}$$

avec P_n^k le pixel d'indice k de la matrice de poids associée à l'image n.



(a) P_1 (b) P_2

FIGURE 4 – les cartes de poids pour I_1 et I_2

3 Construction des cartes de poids final par filtrage guidé

Les cartes de poids P_n sont souvent bruitées et assez mal localisées sur les contours réels des objets de l'image. D'après l'article, utiliser ces matrices de poids dans la formule de fusion donne des images très bruitées avec des créations de nouveaux détails qui n'existaient pas auparavant. Le filtrage guidé des cartes de poids P_n permet d'éviter ces problèmes, et présentent de nombreux atouts que nous présenterons après avoir expliqué son principe.

3.1 Filtrage guidé

Le but du filtrage guidé est de conserver les bords des objets dans l'image. La première hypothèse qui est faite est que l'image de sortie est une transformation linéaire de l'image d'entrée soit

$$O_i = a_k^T * I_i + b_k$$

avec a_k, b_k des vecteurs colonnes de taille 3 et I_i le pixel i avec ces composantes RGB. Cette formule s'applique pour tout les pixels dans une fenêtres de taille (2r+1,2r+1) centrée au pixel k. Le filtre guidé s'appuie sur une image guide P, à partir de laquelle le calcul des coefficients est effectué. La partie compliquée est justement le calcul des coefficients ak, bk. Heureusement l'article nous fournit une formule toute faite :

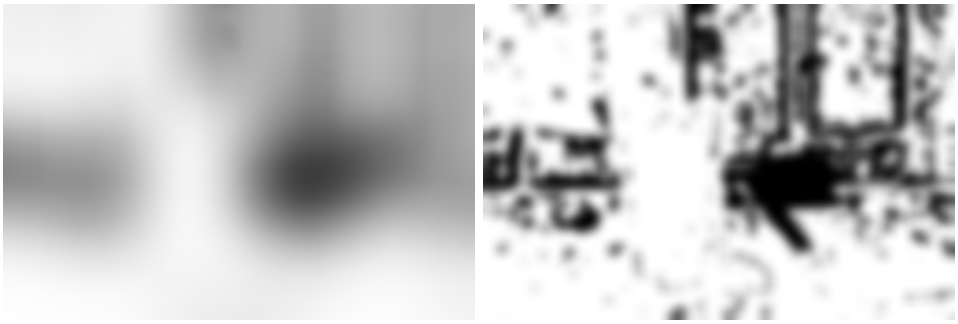
$$\begin{cases} a_k &= (\Sigma_k + \epsilon U)(\frac{1}{|\omega|}\sum_{\omega_k} I_i P_i - \mu_k \overline{P_k}) \\ b_k &= \overline{P_k} - a_k^T \mu_k \end{cases}$$

avec U la matrice identité de taille 3 et Σ la matrice 3x3 de covariance de I dans ω_k.
Le pixel i appartient lui à (2r+1)*(2r+1) fenêtre ω_k. Il n'y a donc pas un seul a_k qui corresponde à ce pixel. C'est pourquoi on calcule la moyenne des a_k et b_k sur la fenêtre ω_i pour obtenir les véritables coefficients a_i et b_i. On obtient donc :

$$O_i = \overline{a_i^T} I_i + \overline{b_i}$$

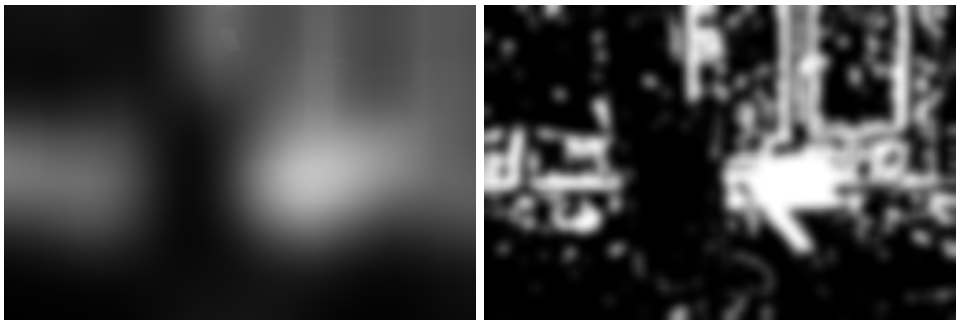
3.2 Calcul des cartes de poids

Comme on l'a dit précédemment les matrices de poids Pn ne conviennent pas dans cette configuration. L'article propose une solution à cela : créer deux nouvelles cartes de poids (une pour le calque de détail et une pour le calque de base). Les cartes sont construites par filtrage guidé de I_n par P_n avec des paramètres ϵ et r, différents pour les deux cartes.



(a) W_1^B (b) $.W_1^D$

FIGURE 5 – les deux nouvelles cartes de poids pour I_1



(a) W_2^B (b) $.W_2^D$

FIGURE 6 – les deux nouvelles cartes de poids pour I_2

4 Reconstruction de l'image à deux échelles

On reconstruit ensuite une image de détail D et une image de base B de cette manière :

$$\overline{B} = \sum_{1,N} W_n^B B_n$$

et

$$\overline{D} = \sum_{1,N} W_n^D D_n$$

On obtient l'image finale en sommant ces deux résultats :

Deuxième partie

Tests et critiques

5 Sur des images fixes

Nous avons implémenté notre algorithme avec le langage de programmation python en utilisant la bibliothèque opencv. Dans un premier temps, nous avons codé la fonction *guidedfilter(im1, im2, r, epsilon)* en parcourant l'image pour calculer les $a_{i,j}$ et les $b_{i,j}$. Ensuite, il fallait recalculer la moyenne des matrices a et b obtenues pour chaque patch. Les moyennes et les variances ont été obtenues en utilisant la bibliothèque numpy. Cette manière de faire est très couteuse en temps, l'algorithme met près de 2 minutes à effectuer les calculs pour des images de taille 672x511. Lorsqu'il faut faire une fusion de 12 images, sachant qu'il faut appliquer deux fois par photo le filtre guidé, le processus prend plus d'une demi-heure à faire la fusion.
Les premiers tests que nous avons effectués ont été sur les douze photos ci-dessous. Ces douze photos ont été prises à différentes profondeurs de champs, de manière à ce que les points nets soient à différents endroits selon chaque photo.
La fusion obtenue n'était pas bonne, nous avons obtenu une image noir et blanche bruitée. Afin

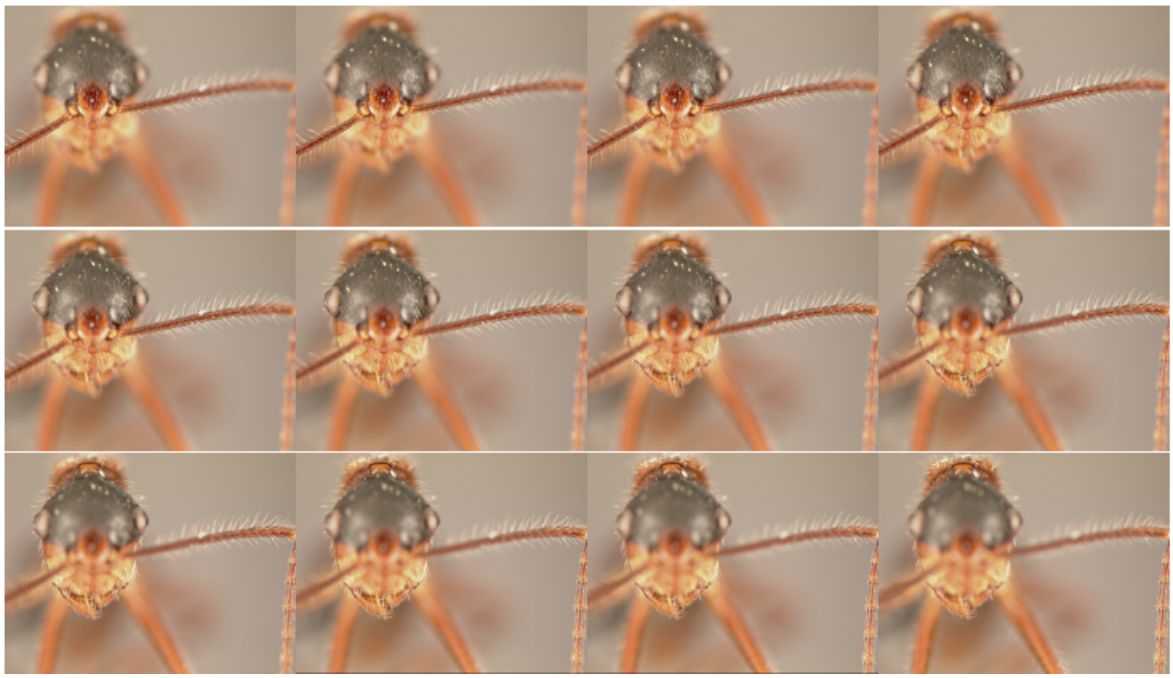


FIGURE 7 – Les douze images de tests

de tester les autres méthodes de l'algorithme, nous avons utilisé une fonction d'opencv qui s'appelle guidedFilter. Nous obtenons le résultat de la Figure 8.

FIGURE 8 – Fusion obtenue avec la fonction guidedFilter d'opencv

L'image obtenue semble, au premier coup d'œil, réussie. On a l'impression qu'elle est nette en tous points. Nous avons donc constaté que les autres méthodes implémentées étaient correctes et que l'erreur précédente provenait de la fonction *guidedfilter*. Tout de même, en regardant de plus près l'image, il y a plusieurs endroits qui ne sont pas satisfaisant. Tout d'abord, la photo a subi un effet de *ringing*.

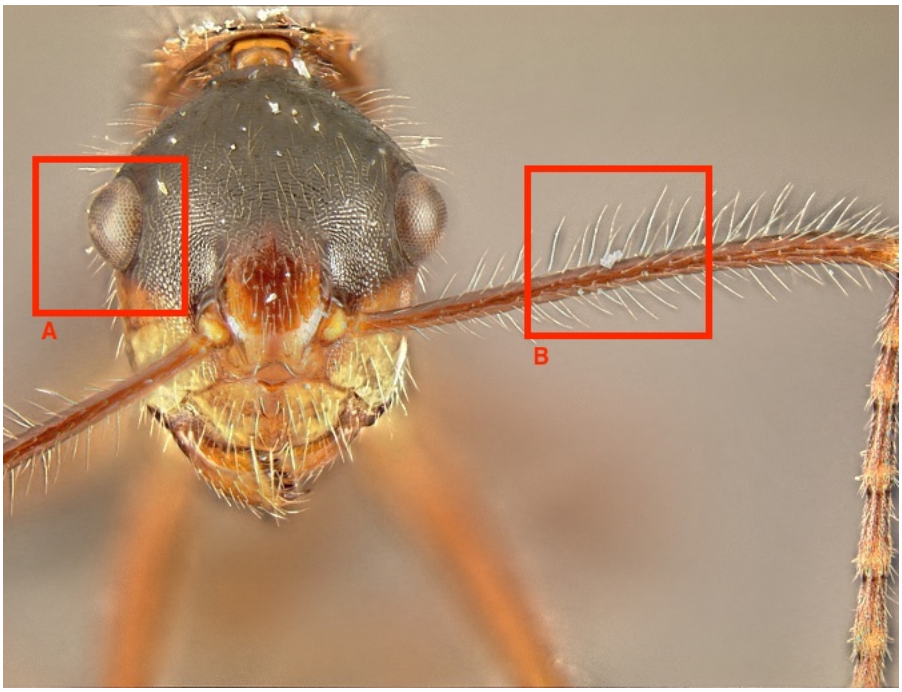


FIGURE 9 – Visualisation des zones de ringing

Dans les deux carrés rouges, ci-dessus, nous pouvons apercevoir les effets de ringing. Dans le carré A, il s'agit de l'œil qui est répété légèrement sur la gauche, créant un effet d'onde. Dans le carré B, ce sont les poils blancs qu'il faut observer. Il y a une ombre qui s'est formée autour de chaque poil. En

comparant avec les douze images originales, et en regardant sur lesquelles d'entre elles ces parties sont nettes, on remarque que l'on devrait obtenir des résultats plus proches de cela :

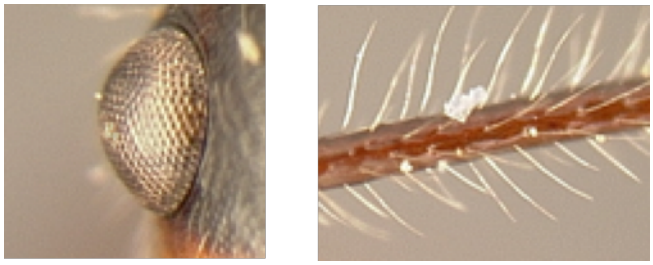


FIGURE 10 – Images extraites des photos originales

Sur les deux images, il n'y a pas l'effet de répétition des formes. Il s'agit donc bien de l'algorithme qui multiplie les formes.
Un autre de ses effets est qu'il modifie la couleur de certains pixels.

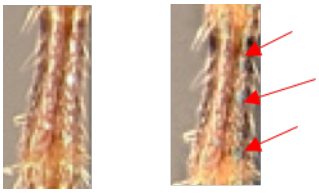


FIGURE 11 – À gauche : image d'origine. À droite : image obtenue après fusion

On remarque que l'image de droite comporte des zones qui ont été assombries. De plus en trois points (ceux marqués par les flèches), il y a une apparition de points bleus sur l'image. À travers cet exemple, nous avons compris que la fonction donnée par opencv ne permet pas d'obtenir une fusion d'images convenable pour ce que nous voulons faire. Nous avons donc implémenté la fonction guidedfilter autrement (appelée *guidedfilterbis* dans le code). Plutôt que de parcourir l'image pixel par pixel, nous avons procédé par convolution. En effet, les opérations du filtre guidées ne constituent principalement qu'en des calculs de moyennes et de variances. Nous avons donc calculé celles-ci en convolant par des filtres moyens et en les multipliant terme à terme entre-elles. L'algorithme ne parcours l'image qu'en une seule occasion : lors du produit matriciel pour le calcul de $a_{i,j}$ dans le cas d'images de couleurs. Nous n'avons pas trouvé le moyen d'éviter ce parcours de l'image. Tout de même, le temps de calcul est nettement inférieur à celui de notre fonction originale : la fusion des douze images ne prend plus que 2 minutes.
L'image obtenue est celle-ci :

FIGURE 12 – Image obtenue par fusion avec la fonction guidedfilterbis

Le résultat est, pour la plupart des aspects, meilleur que la fusion précédente. Tout d'abord, on n'observe plus de ringing. Dans les zones proches des yeux ou des poils blancs, il n'y a plus l'apparition d'ombre noire. Ensuite, nous n'observons pas de points bleus sur l'image qui seraient causés par la fusion des images.
Pourtant, il existe une zone où le résultat est moins bon avec cette fonction. Il s'agit des jambes. Il n'y a aucune des douze images d'origines où elles sont nettes. Ici, elles sont complètement floues. Or, sur la fusion précédente elles étaient un peu plus nettes.

FIGURE 13 – À gauche : l'image 12 de la base de donnée d'origine, à droite : image obtenue avec la fonction guidedfilterbis

En regardant le contenu des deux carrés rouges, on constate que les jambes de la fourmi obtenue par fusion sont trop floues, elles auraient dû être plus nettes.

6 Avec du mouvement sur l'image

Dans l'article dont est issu l'algorithme testé, un exemple a capté notre attention. Il s'agit de celui illustré sur les photos ci-dessous.

(b1) Source 1 (b2) Source 2

Dans l'image Source 1, seule la fille est nette et dans l'image source 2, seule la pierre au premier plan est nette mais la fille s'est déplacée. L'algorithme retourne une image où la fille et la pierre sont floues. Malgré le déplacement, il n'y a pas eu de problème de superposition. Le coin inférieur gauche est particulièrement intéressant. Sur l'image 2, on y voit un bout de la manche de la fille, qui est flou et sur l'image 1 on y voit le fond de l'image. En fusionnant les deux images, on ne voit plus du tout la manche mais juste le fond de l'image 1.
Nous avons donc pris deux photos similaires où nous déplaçons un objet entre les deux images et où nous changeons la mise au point.

Nous obtenons :

FIGURE 14 – Image de la fusion

Nous voyons que le verre sur la première image n'a pas disparu, il a été moyenné avec l'autre image, ce qui lui donne une impression de transparence. La différence entre cette image et la manche de la fille

sur l'exemple précédent vient peut-être du fait que le fond de l'image sans la manche est relativement net. L'application du laplacien a révélé plusieurs contours sur cette zone et donc, finalement, elle a été sélectionné en priorité par rapport à la manche de l'autre image. En ce qui concerne le verre, même s'il était flou sur la première image, le contraste élevé de couleur a permis au laplacien de retourner des valeurs élevées dans cette zone et donc que l'image des poids après avoir été filtrée ne soit pas nulle dans cette même zone.

Troisième partie

Conclusion

Finalement, l'algorithme est efficace lorsqu'il s'agit de fusionner des images qui ont été prises en photo simultanément. C'est à dire qu'il n'y a pas eu de mouvement d'objet entre la prise des différentes photos. Cet méthode devient inefficace lorsqu'il y a un déplacement d'un objets, c'est ce que nous avons vu dans la partie précédente. Toue de même, certains cas particuliers existent, notamment lorsque l'objet net de la première photo est placé devant le même objet, qui n'est plus net, de la seconde photo. Cet algorithme pourrait avoir une application dans la photographie pr smartphone. Cela pourrait être une technique pour augmenter sa profondeur de champ artificiellement. L'appareil prendrait alors plusieurs photos en même temps, chacune faisant la mise au point en différents endroit et, avec cet algorithme, le smartphone fusionnerait les images pour n'en former qu'une seule. L'obstacle principal à sa mise en oeuvre est le temps de calcul. La principale difficulté pour l'optimisation de l'algorithme vient probablement de la construction des matrice de poids. En effet, la recherche du maximum pour chaque pixel impose de parcourir les images pixel par pixel.

Table des matières