

Reconstruction d'un tableau sans reflets à partir de photos prises sous des angles différents

Théophile DALENS Jean CAILLÉ Henri MAITRE (encadrant)

16 Mai 2014

1 Introduction

Il devient aujourd’hui facile d’obtenir des photos de chefs-d’œuvre dans un musée. Toutefois la présence de lumières, couplée aux vernis et aux cadres en verre, entraîne sur la plupart des photos prises des reflets peu plaisants à l’œil. En multipliant les points de vue, nous allons voir qu’il est possible de supprimer ces reflets et d’obtenir une photo proche de celle prise dans des conditions idéales.

2 Méthode

La méthode proposée par l’article [3] ne s’attache pas à traiter le tableau sous l’angle de la reconnaissance d’image (reconnaitre par exemple les aplats de couleurs et supprimer les reflets à partir de cette information), mais uniquement grâce à des techniques génériques, permettant leur application sur la plupart des tableaux. L’idée consiste à prendre plusieurs photos du même tableau sous différents angles. Les reflets ne dégraderont pas la même zone du tableau sur les différentes images et nous pourrons reconstruire à partir de ces images une image “virtuelle” sans reflets.

L’algorithme suggéré par l’article se décompose en deux parties. Tout d’abord, les images sont ”recalées” afin d’obtenir une perspective commune. Ainsi, les pixels des différentes images correspondent aux mêmes zones sur les tableaux, et il devient plus facile de reconstruire l’image corrigée. La seconde partie correspond justement à cette reconstruction. Plusieurs méthodes sont suggérées et nous nous sommes attachés à implémenter et comparer les différentes méthodes de fusion d’image proposées.

2.1 Recalage des images

Comme décrit plus haut, la première partie revient à ”recaler” les images entre elles, c’est à dire simuler une perspective commune entre les photos. Il n’est pas possible de modifier toutes les photos pour recalier l’image du tableau sur un rectangle donné car cela nécessiterait une connaissance *a priori* des dimensions du tableau, que nous ne connaissons pas. Pour palier à ce problème, nous choisissons parmi les images de la peinture une photo dite de *référence* et nous recalons les autres pour les afficher sous cette perspective.

Dans le cas où la caméra est idéale (modèle sténopé, pas de déformation dues à la lentille), et le tableau est parfaitement plan, la transformation permettant de recaler les images est une perspective (homographie). Nous n'avons pas eu besoin lors de nos traitements d'hypothèses supplémentaires.

2.1.1 Recalage manuel

Pour trouver la perspective liant deux images données, la solution la plus simple est de demander à l'utilisateur de cliquer sur 4 points communs entre les deux photos (par exemple les quatre coins du tableau). On obtient alors 2 quadrilatères quelconques, définis par les points $(p_i)_{i=1\dots 4}$ dans l'image de référence et $(q_i)_{i=1\dots 4}$ dans l'image que l'on souhaite recaler.

On peut à partir de ces points trouver une perspective permettant de passer du premier quadrilatère à l'autre, en effet, il existe en effet une et unique matrice M en coordonnées homogènes tel que

$$\forall i \in \{1\dots 4\}, q_i = Mp_i$$

Une fois que nous avons trouvé la matrice M , il est possible de redresser l'image pour que les deux quadrilatères coïncident, en appliquant la perspective définie par M .

Toutefois, il est peu probable que les valeurs de l'image de référence à échantillonner aient des coordonnées entières (*e.g.* des pixels). Nous devons donc pouvoir interpoler l'image de référence pour construire l'image redressée. Plusieurs méthodes sont alors possibles :

Interpolation au plus proche Pour trouver la valeur du pixel (x, y) dans l'image de référence, on arrondit simplement les deux nombres pour tomber sur un pixel. Cette méthode est simple à utiliser, mais moins précise que l'interpolation linéaire.

Interpolation Bilinéaire Dans cette méthode, pour trouver la valeur du pixel (x, y) , on observe les 4 valeur des pixels les plus proches et on interpole bilinéairement.

En pratique, nous avons choisi d'interpoler l'image de façon bilinéaire, plus précise et donc plus adaptée au travail de fusion qui viendra par la suite.

Problème avec la méthode manuelle En pratique, cette méthode de recalage manuelle n'est pas utilisable pour plusieurs raisons. D'une part, le temps nécessaire pour définir les coins de chaque peinture est relativement long. D'autre part, la précision obtenue lors de nos essais n'est pas suffisante pour l'étape suivante du traitement. En effet, l'utilisateur ne peut cliquer qu'avec une précision de quelques pixels au plus. Pour palier à ces problèmes, l'article suggère une solution plus automatisée, faisant appel à des descripteurs locaux.

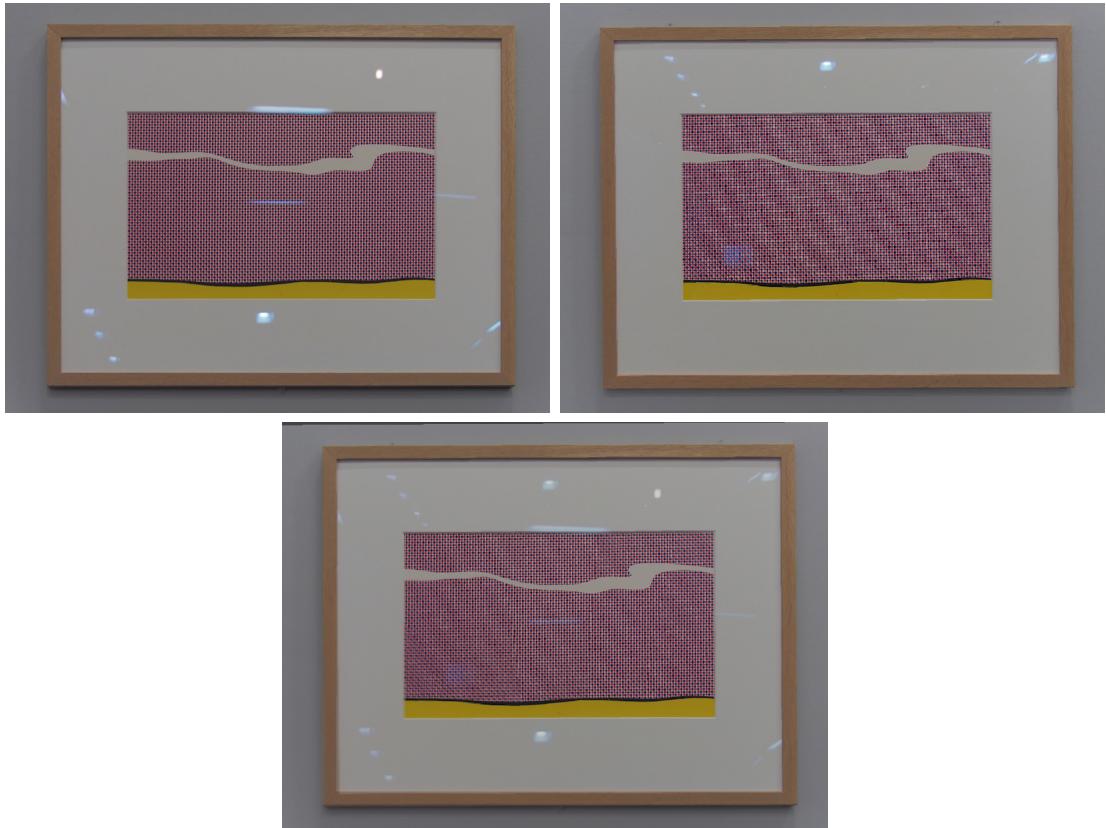


Figure 1: En haut : Image de référence et image après recalage avec la méthode manuelle. En bas : Superposition des deux images. On observe des zones (particulièrement en bas à gauche) où le recalage n'est pas bon.

2.1.2 Recalage automatique avec utilisation des SIFT

Pour obtenir une précision sous-pixellique, nous utilisons des descripteurs locaux pour trouver des correspondances entre les deux images.

SIFT Les SIFT (Scale Invariant Feature Transform, [4]) sont des descripteurs locaux globalement invariants aux transformations linéaires. Ils sont robustes aux transformations homographiques si la perspective n'est pas trop importante. Ils sont généralement utilisés pour détecter les correspondances entre des objets présent dans deux images différentes et sont donc très adaptés au problème actuel. Ils sont de plus invariants aux changements de luminosité et de contraste, les rendant plus robustes au problème de recalage.

Recherche de correspondances Nous extrayons de chaque image plusieurs centaines de SIFTS, auxquels sont associés leurs descripteurs. Nous recherchons ensuite des correspondances entre les images. Plusieurs algorithmes existent, mais le plus simple consiste à prendre pour chaque point de la première image le point le plus proche (dans l'espace des descripteurs) comme correspondant. Pour éviter d'obtenir un trop grand nombre de faux positifs, un seuillage est effectué, nous permettant d'obtenir des correspondances point-à-point entre les deux images:



Figure 2: Paires de descripteurs SIFT entre les deux images. Il existe encore un grand nombre de fausses correspondances

RANSAC Il reste donc à trouver la matrice M permettant de passer de l'image à recaler à l'image de référence. Le problème est différent du recalage manuel car il n'existe pas de telle matrice pour plus de quatre paires de points en général. Toutefois, si toutes les correspondances sont correctes (et il n'existe pas de faux positifs), il est possible de trouver une matrice minimisant l'erreur e , par exemple par descente de gradient.

$$e = \sum_{i=1}^n \|p_i - Mq_i\|^2$$

Cependant, en présence de faux positifs (c'est-à-dire des paires de points qui ne correspondent pas d'une image à l'autre), il est nécessaire d'employer un algorithme plus robuste pour trouver cette homographie. La méthode RANSAC [2] permet à la fois de filtrer les bonnes correspondances et d'obtenir cette homographie. Cet algorithme itératif sélectionne un sous-ensemble de descripteurs aléatoirement et marque ces correspondances comme "bonnes". On cherche ensuite l'homographie la plus adaptée à cet ensemble de point (en minimisant l'erreur e par exemple), puis on cherche les correspondances qui sont "d'accord" avec cette homographie (celles pour lesquelles l'erreur résiduelle est inférieure à un certain seuil). Si suffisamment de paires sont en accord avec l'homographie trouvée, elle est conservée, sinon, l'algorithme est relancé avec une nouvelle initialisation aléatoire.

Cet algorithme est extrêmement robuste et peut supporter un grand nombre de faux-positifs (plus de 50%). Toutefois, la convergence n'est pas forcément assurée. Il n'y a également un certains nombres de paramètres dont les valeurs sont à ajuster en fonction du problèmes.

En pratique toutefois, et sur la majorité des photographies, l'algorithme de RANSAC permet de trouver une bonne correspondance entre les différentes images.

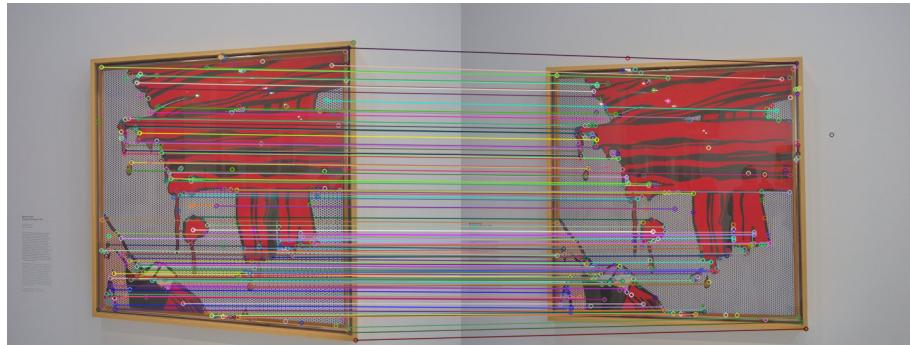


Figure 3: Correspondances de descripteurs SIFT estimées correctes par l'algorithme RANSAC et utilisées pour reconstruire la perspective



Figure 4: A gauche : image de travail, au centre : image de référence, à droite : image de travail après recalage par rapport à l'image de référence. Noter la différence de position des reflets entre l'image de référence et l'image recalée

2.1.3 Améliorations possibles

Plusieurs améliorations des méthodes proposées auparavant sont possibles, tant sur le modèle que sur la robustesse du système et du traitement des données.

Prise en compte des déformations dues à la lentilles La méthode proposée ci-dessus n'est pas parfaite. En effet, les déformations dues à la lentille de la caméra induisent des imperfections dans les images. Une simple perspective ne peut pas corriger ces déformations, et le recalage n'est pas idéal.



Figure 5: Détails de la moyenne de l'image recalées et de l'image de référence. A gauche, le recalage est correcte et les contours sont nets. A droite, sur le bord de l'image, le recalage est moins bon, et les contours sont flous.

Pour palier à ce problème, l'article suggère un modèle prenant en compte les déformations dues à la lentille, et se basant sur un polynôme de degré 4 $P(x, y)$. Pour trouver ce polynôme, la matrice de perspective M est obtenue par l'algorithme de RANSAC, puis les points en "accord" avec cette matrice au sens de RANSAC sont utilisés pour trouver les meilleurs coefficients du polynôme. La correction à appliquer à l'image est donc

$$q'_i = Mq_i + P(q_i)$$

Amélioration de la robustesse de la méthode SIFT Sur plusieurs jeux d'images, la méthode SIFT n'a pas été adaptée, et proposait des recalages aberrants, inutilisable pour la fusion d'image.

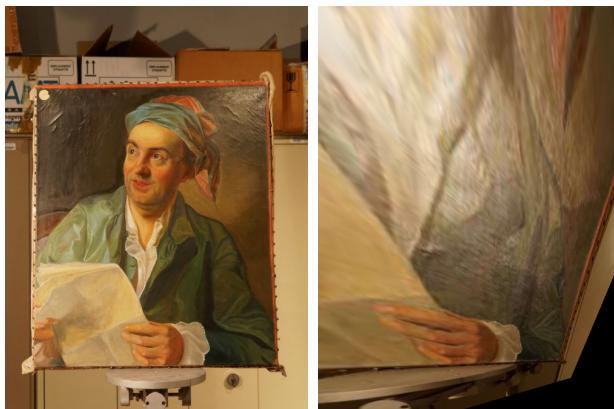


Figure 6: Image de référence, et image incorrectement recalés

Deux solutions possibles peuvent venir à l'esprit. D'une part, nous pourrions rejeter automatiquement les images où le nombre de correspondances "bonnes" selon RANSAC est trop faible, et où le recalage n'est pas acceptable. D'autre part, nous pourrions demander à l'utilisateur d'indiquer grossièrement les coins du tableau. Ceci permet d'obtenir

une approximation de la "bonne" perspective, et de restreindre la recherche par RANSAC à un espace plus petit.



Figure 7: Moyennes des images recalées sans rejet (toutes les images recalées sont gardées) et avec rejet (les images où le nombre d'inlier est trop faible est écarté)

2.2 Fusion des images

Une fois les images recalées, l'étape suivante est de fusionner les images entre elles pour éliminer les reflets. Plusieurs approches sont possibles ; dans les premières approches développées ci-après, nous testons des approches qui fusionnent les images pixels par pixels, c'est-à-dire que chaque pixel de l'image reconstitué dépend uniquement des pixels correspondant dans les différentes images recalées. Ainsi, si les images recalées sont notées $\tilde{I}_1, \dots, \tilde{I}_n$, l'image fusionnée J est définie par

$$J(x, y) = f(\tilde{I}_1(x, y), \dots, \tilde{I}_n(x, y))$$

De nombreux choix sont possibles pour la fonction f . Une première tentative a été de déterminer quels étaient les points des images qui devaient être corrigés, c'est-à-dire les points qui présentaient des reflets. Pour ce faire, nous avons tracé les courbes d'intensités comparées de deux images recalées (figure 8). Ces courbes d'intensité montrent la complexité du problème ; même recalées, les images présentent des variations locales importantes d'intensité (la droite a une forte épaisseur). Ainsi, il est difficile de catégoriser les points présentant des reflets. La fonction f doit donc prendre en compte tous les points. La moyenne arithmétique ne permet pas d'obtenir une fusion efficace ; elle atténue les reflets mais les multiplie (en nombre) sur l'image fusionnée.

Dans la section 2.2.1, nous proposons une fusion par minimum sur chaque canal de couleur. Nous proposons ensuite des fusions par médiane ; elle se fait par canal RGB dans la section 2.2.2, par médiane vectorielle dans la section 2.2.3, ou en passant dans l'espace de

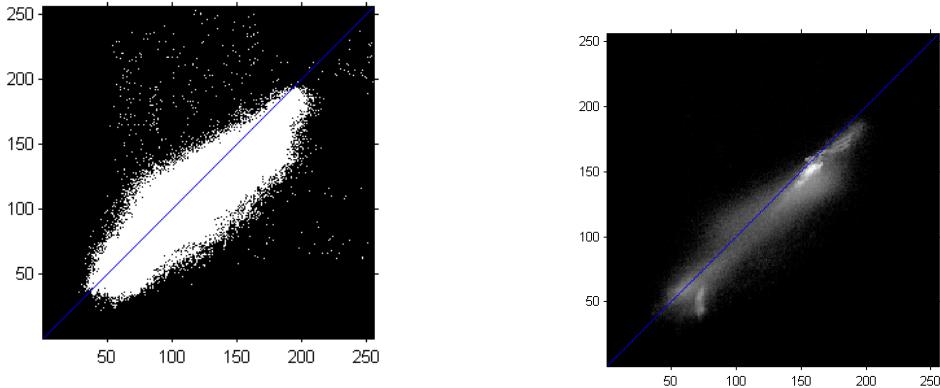


Figure 8: Courbes d'intensités comparées. Sur l'image de gauche, chaque point (x, y) blanc correspond à l'existence d'un couple dont l'intensité est x sur une image et y sur l'autre. A droite, le nombre de points correspondant est sommé et passé en échelle logarithmique. En bleu, la droite d'équation $y = x$. Les points s'éloignant fortement de cet axe sont probablement des points présentant des reflets.

couleur CIELAB dans la section 2.2.4.

Il est également possible de travailler dans l'espace gradient au lieu de l'espace de l'image. Dans ce cas, l'image fusionnée J est définie par

$$\nabla J(x, y) = f \left(\nabla \tilde{I}_1(x, y), \dots, \nabla \tilde{I}_n(x, y) \right)$$

Ce cas est détaillé dans la section 2.2.5. Les résultats obtenus par ces méthodes sont comparées dans la section 4.

2.2.1 Fusion par minimum

Pour cette méthode, nous faisons l'hypothèse que les reflets ne font qu'ajouter de la luminosité aux différents pixels de l'image, et que les reflets ne peuvent se retrouver dans toutes les images. Dès lors, il suffit de prendre le minimum de chaque canal de couleur pour parvenir à éliminer les reflets. En notant R , G et B les canaux de couleurs, on écrit

$$J(x, y, c) = \min_{i=1..N} \tilde{I}_i(x, y, c) \quad \forall c \in \{R, G, B\}$$

Cette méthode, bien que simple, élimine très efficacement les reflets lambertiens. Elle présente cependant le désavantage de donner des couleurs non naturelles, c'est-à-dire qu'elle ne garantit pas que les pixels de l'image J ainsi obtenue proviennent des images d'origine. De plus, il est attendu que cette méthode assombrisse les images ; en particulier, d'éventuelles ombres se verront assez nettement avec cette méthode.

2.2.2 Fusion par médiane car canal

Un autre choix possible pour la fonction est la médiane :

$$J(x, y, c) = \text{median}_{i=1..N} \tilde{I}_i(x, y, c) \quad \forall c \in \{R, G, B\}$$

Ici, l'hypothèse faite est que les reflets se retrouvent sur moins de la moitié des pixels considérés. Elle est donc plus forte que l'hypothèse de la section précédente ; en particulier,

cette méthode est inadaptée si une majorité des prises de vue sont semblables. Cependant, elle est plus robuste à des données très différentes des autres.

2.2.3 Fusion par médiane vectorielle

La méthode précédente présente le désavantage de mélanger les canaux couleurs d'images différentes, ce qui peut conduire à des couleurs non naturelles. Pour pallier à ce problème, nous avons également implémenté un équivalent de la médiane dans l'espace vectoriel des couleurs, appelée "médiane vectorielle". En notant $(\mathbf{x}_i)_{i \in I}$ une liste de vecteurs, elle est définie par

$$\text{vecmedian}((\mathbf{x}_i)_{i \in I}) = \underset{i \in I}{\operatorname{argmin}} \sum_{j \in I} \|\mathbf{x}_i - \mathbf{x}_j\|_1$$

La norme L_1 est ici utilisée. Il est possible de vérifier que cette médiane est identique à la médiane classique lorsque le nombre de dimensions est 1.

Ainsi, pour la fonction f que nous choisissons est ici la fonction vecteur median :

$$J(x, y) = \text{vecmedian}\left(\tilde{I}_1(x, y), \dots, \tilde{I}_n(x, y)\right)$$

En toute rigueur, le vecteur médian n'est pas toujours bien défini. Il est possible en effet que plusieurs vecteurs \mathbf{x} atteignent le minimum de la définition. En pratique, nous prenons soin de toujours sélectionner des vecteurs provenant de la même image en cas de conflit entre deux images ; ainsi, il n'y a pas de discontinuité dans l'image résultante liée à ce problème. Dans le cadre de la médiane canal par canal, si le nombre d'images est pair, nous utilisons la moyenne des deux scalaires pouvant convenir.

2.2.4 Fusion par médiane dans l'espace CIELAB

Il est possible de changer l'espace de couleur avant de procéder à la médiane. Ainsi, pour cette méthode, nous convertissons nos images dans l'espace de couleur CIELAB (CIE signifie "Comission Internationale de l'Éclairage"). Dans cet espace, le L correspond à la luminosité ; a et b indiquent la position de la couleur sur des axes vert/magenta et bleu/jaune. Cet espace est réputé plus uniforme perceptuellement que l'espace RGB. Ainsi, deux couleurs très proches à l'œil seront très proches dans cet espace. Une fois la conversion faite, nous utilisons la médiane composante par composante.

En pratique, les différentes médianes présentées changent peu les résultats.

2.2.5 Fusion dans l'espace gradient

Enfin, une méthode décrite dans [3] permettant de réduire plus les reflets consiste à prendre la médiane vectorielle dans l'espace gradient des images. Dans un premier temps, chaque image recalée \tilde{I} est convertie en niveau de gris \tilde{I}^g . Nous calculons ensuite le gradient de chacune de ces images. Nous en construisons une carte d'index, définie par

$$a(i, j) = \underset{k}{\operatorname{argvecmedian}} \nabla \tilde{I}_k^g(i, j)$$

Nous prenons donc les indices des images correspondant au vecteur médian en chaque point. Cette carte (voir figure 9) nous sert à construire le gradient de l'image résultante en chaque canal :

$$F(x, y, c) = \tilde{\nabla} I_{a(i, j)}(x, y, c)$$

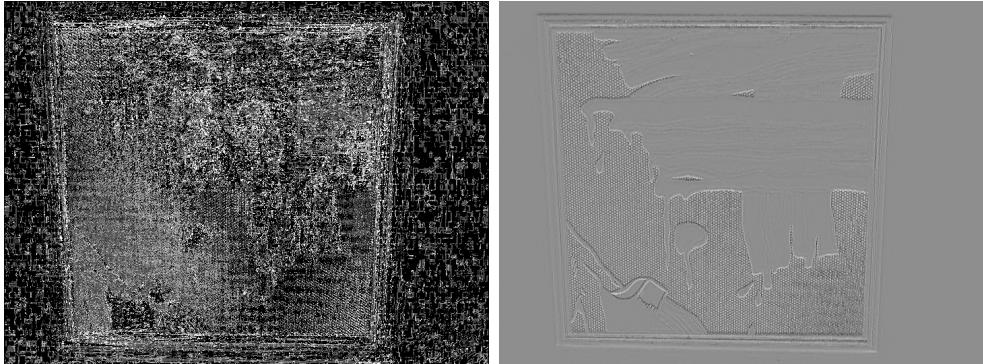


Figure 9: Laplacien de l'image à reconstruire. A gauche, les index à partir duquel sont calculés l'image gradient F . Chaque niveau de gris différent correspond à un index différent. A droite, le laplacien $\text{div}F$ sur le canal de couleur B.

Pour chaque canal de couleur c , nous reconstruisons l'image J qui devrait être telle que

$$\nabla J(x, y, c) = F(x, y, c)$$

Cependant, il est important de noter ici que F n'est pas réellement le gradient d'une image, et, par conséquent, il est impossible de résoudre cette équation. Nous nous efforcerons de fait de minimiser

$$||\nabla J - F||$$

L'équation de Poisson s'écrit

$$\Delta J = \text{div}F$$

c'est-à-dire, en chaque point,

$$J(i+1, j) + J(i-1, j) + J(i, j+1) + J(i, j-1) - 4J(i, j) = F(i, j)$$

Pour la résoudre, nous procédons par une méthode itérative. Nous initialisons J par

$$J_0 = \tilde{I}_1$$

et nous répétons, jusqu'à convergence, l'équation

$$J_{n+1}(i, j) = \frac{1}{4} (J_n(i+1, j) + J_n(i-1, j) + J_n(i, j+1) + J_n(i, j-1) - F(i, j))$$

En pratique, une douzaine d'itérations suffit pour éliminer les reflets. Cependant, le système ne converge pas car $\text{div}F$ n'est pas réellement un Laplacien. De fait, nous observons des zones d'instabilité assez marquées, qui détruisent progressivement l'image. Ces zones naissent plus rapidement aux endroits où le recalage d'images est perfectible. De fait, cette méthode ne donne pas des résultats comparables avec nos autres méthodes (voir figure 10).

3 Implémentation

3.1 Choix techniques

Nous avons implémenté la méthode décrite dans l'article à l'aide du langage C++ accompagné de la librairie OpenCV [1] de traitement d'image. Cette librairie, très utilisée dans



Figure 10: A gauche, différence entre deux images recalées. On voit assez nettement de grandes différences sur le bord bas droit de la peinture. Cela peut être du à un recalage imparfait. A droite, image fusionnée obtenue par la méthode des gradients médians. On aperçoit des zones saturées (blanches et noires) qui correspondent à des divergences dans la résolution de l'équation de Poisson. Ces zones correspondent aux zones où le recalage est imparfait.

l'industrie, permet d'accélérer le traitement d'image et met à disposition de l'utilisateur plusieurs fonctions (comme par exemple l'algorithme de RANSAC, implémenté dans la fonction `findHomography`).

3.2 Architecture

Les méthodes décrites ci-dessus est fortement dépendante des paramètres d'entrées et des images à traiter. Par exemple, sur certaines images, la méthode de fusion par minimum est plus adaptée que la méthode de fusion par médiane. Il est donc nécessaire de permettre à l'utilisateur de sélectionner les différentes techniques de recalage et de fusion d'image. Ainsi, nous avons séparé les différentes responsabilités entre plusieurs fichiers et classes. Plusieurs méthodes(situées dans les fichiers `ImageFitter` `ImageMerger`) permettent de sélectionner les différentes techniques possibles.

3.3 Collaboration

La collaboration a été réalisé grâce au gestionnaire de version git. La totalité du code et ses révisions antérieures sont donc disponible sur Github : <http://github.com/jcaille/CVCanvas>

4 Résultats

Nous présentons nos résultats sur 3 jeux de photographies de peintures dont nous donnons le tableau de référence (celui qui sert à recaler les images) figure 11. Toutes les images sont disponibles dans le dossier Images de notre répertoire de travail Github. Les résultats varient légèrement selon les méthodes. La méthode du minimum élimine le mieux les reflets (figure 12). Elle présente également l'avantage d'être la seule à fonctionner avec seulement deux photographies. Cependant, elle est par nature très sensible aux données initiales. Ainsi, si le recalage n'est pas bon, cette méthode ne marche pas du tout. Ainsi, il est nécessaire d'ajouter un seuil pour sélectionner les images pouvant être fusionnées.

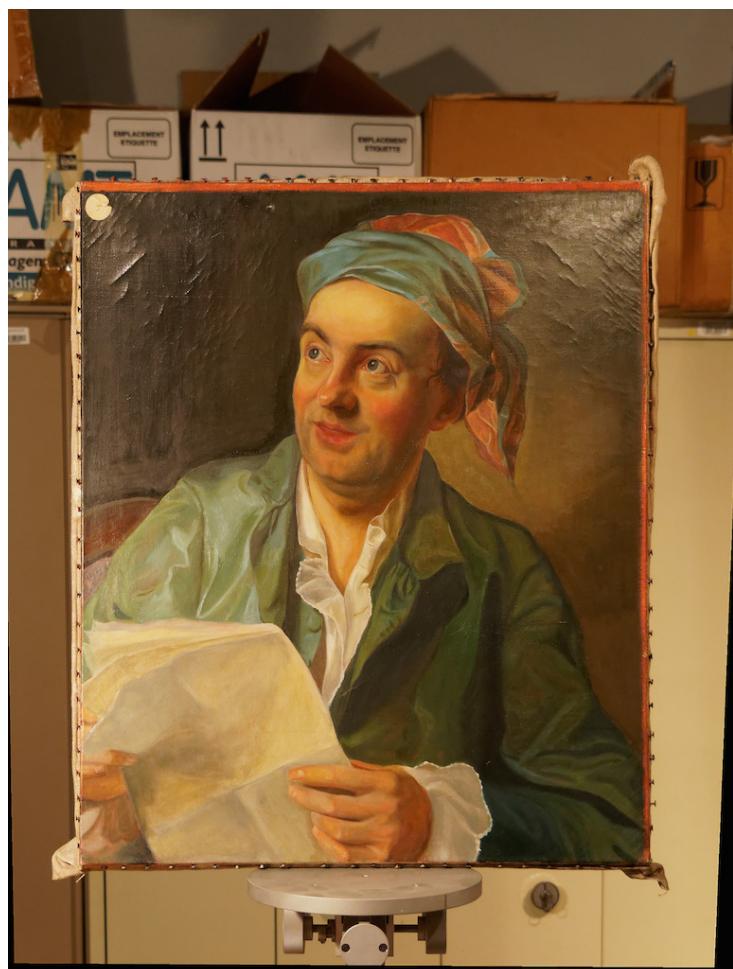


Figure 11: images d'origine. Dans cette section nous travaillons avec respectivement 10, 4 et 2 photographies de chaque peinture.



Figure 12: Elimination des reflets avec la méthode du minimum. Les reflets lambertiens sont très bien éliminés.

Si le recalage a un nombre de points de correspondances en dessous du seuil, elle n'est pas prise en compte. Les résultats s'améliorent (figure 13).

Les méthodes de médiane présentent l'avantage d'être plus robustes à d'éventuels mauvais recalages. Elles éliminent cependant moins bien les reflets. Les résultats sont très proches (figures 14 et 15). La médiane canal par canal élimine le mieux les reflets. Nous n'observons pas de problème spécifique avec cette méthode ; les couleurs apparaissent toujours naturelles. Il reste cependant plus de reflet que la méthode par minimum.

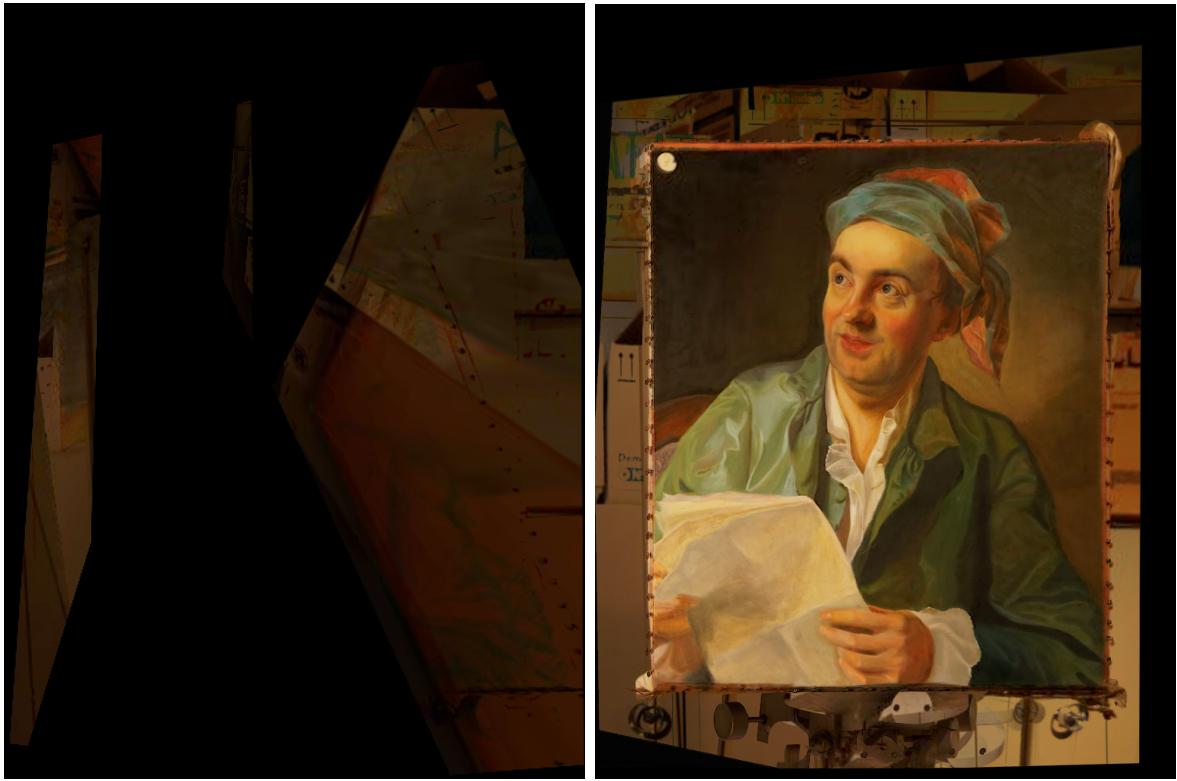


Figure 13: Méthode du minimum. A gauche, une image mal recalée fait échouer la fusion. A droite, seules les images bien recalées sont conservées. Bien que les reflets soient bien retirés, on observe une perte de détails sur les plis du manteau.

5 Améliorations possibles

Plusieurs améliorations de l'algorithme sont envisageables. Si les images obtenues sont de bonne qualité, l'algorithme ne sait pas faire la différence entre les zones de la peinture et les zones extérieures (cadre, mur, paysage, ...), dans lesquelles la fusion est effectuée alors qu'elle n'a pas de sens. Une possibilité pour détecter ces zones et de segmenter l'image de référence (par exemple en utilisant des snakes quadrangulaires) entre un "intérieur" sur lequel la fusion serait effectué et un extérieur sur lequel on garderait le contenu d'origine. Pour s'aider dans cette segmentation, nous pourrions utiliser les informations des autres photos du tableau.

De plus, l'algorithme que nous avons développé (en particulier la partie de fusion) n'est pas de fait résistant aux changements de contraste en général. Une idée pour corriger ce défaut serait d'égaliser les histogrammes les différentes images (de préférence en limitant cette égalisation à la zone "intérieur" du tableau).

References

- [1] G. Bradski. *Dr. Dobb's Journal of Software Tools*.
- [2] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

- [3] Gloria Haro, Antoni Buades, and Jean-Michel Morel. Photographing paintings by image fusion. *SIAM Journal on Imaging Sciences*, 5(3):1055–1087, 2012.
- [4] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.



Figure 14: Fusion par médiane. Haut : médiane canal par canal. Bas : méthode minimum (pour comparaison). Les autres médianes sont identiques à l'œil à la première sur cet exemple. Plus de reflets persistent comparativement à la méthode du minimum (lampes en haut de l'image).



Figure 15: Fusion par médiane. Haut : médiane canal par canal, médiane vectorielle. Bas : dans l'espace CIELAB, dans l'espace gradient. Les détails sont mieux conservés que la méthode par minimum pour les trois premiers. La médiane canal par canal semble obtenir le résultat présentant le moins de reflets.