

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

Théophile DALENS et Jean CAILLÉ

13 Mars 2014

1 Introduction

Il devient aujourd’hui facile d’obtenir des photos de chefs-d’œuvre dans un musée. Toutefois la présence de lumières couplées aux vernis et aux cadre en verres 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 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 pluspart des tableaux. L’idée consiste à prendre plusieurs photos du même tableau sous différents angles. Les reflets ne dégraderont pas alors 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 le tableau, 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.

FIGURE - 3 ETAPES

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. Soit nous pouvons nous attacher à modifier toutes les photos pour recaler l’image du tableau sur un rectangle donné, mais cela nécessite 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 recadrerons 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 un tableau est une perspective. Nous n'avons pas eu besoin lors de nos traitement 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 $p_{i=1\dots 4}$ dans l'image de référence et $p_{i=1\dots 4}$ 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 coïncide, 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 possible :

Interpolation au plus proche Pour trouver la valeur du pixel (x, y) dans l'image de référence, on arrondi simplement les deux nombres pour tomber su 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 test 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.

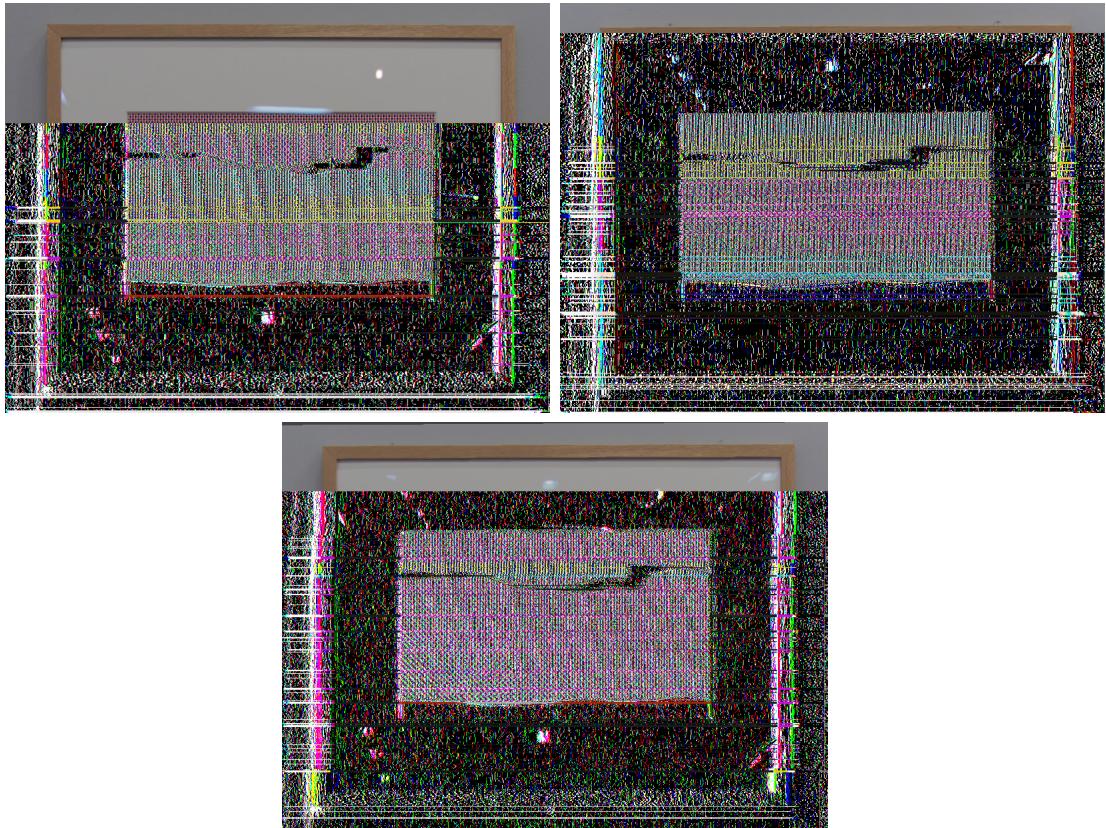


Figure 1: En haut : Image de référence et image après recalage avec la méthode manuelle. En bas : Superposition des deux images. On observe des zones (particulièrement en bas à gauche) où le recalage n'est pas bon.

2.1.2 Recalage automatique avec utilisation des SIFT

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

SIFT Les SIFT [?](Scale Invariant Feature Transform) sont des descripteurs locaux globalement invariants aux changements de perspectives. 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 d'échelles et de luminosité, rendant plus robuste l'algorithme de recadrage.

FIGURE

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

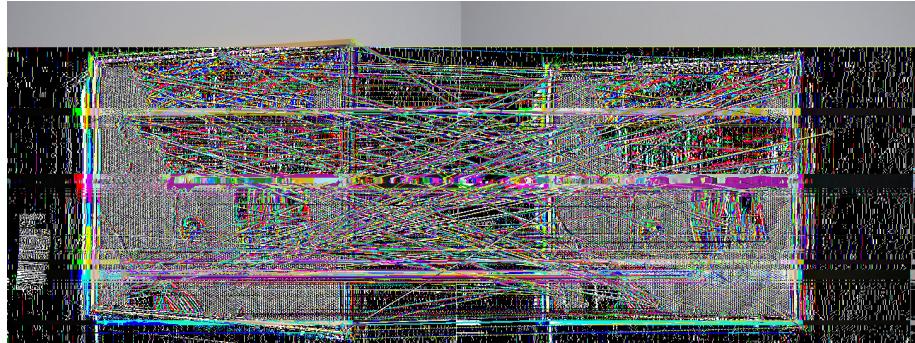


Figure 2: Paires de descripteurs SIFT entre les deux images. Il existe encore un grand nombre de fausses-correspondances

RANSAC Il reste donc à trouver la matrice M permettant de passer de l'image à recadre à l'image de référence. Contrairement au recalage manuel, il n'existe pas de tel matrice pour plus de quatre paires de points. 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, la présence de faux-positifs (*i.e.* 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 [?] 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 (celle pour lesquelles l'erreur résiduelle est inférieure à un certain seuil). Si suffisement 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 (jusqu'à 50% dans certains cas). Toutefois, la convergence n'est pas forcément assurée. Il y'a également un certains nombres de paramètres dont les valeurs sont à ajuster en fonction du problèmes.

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



Figure 3: Paires de descripteurs SIFT estimés correctes par l'algorithme RANSAC et utilisés pour renconstruire la perspective

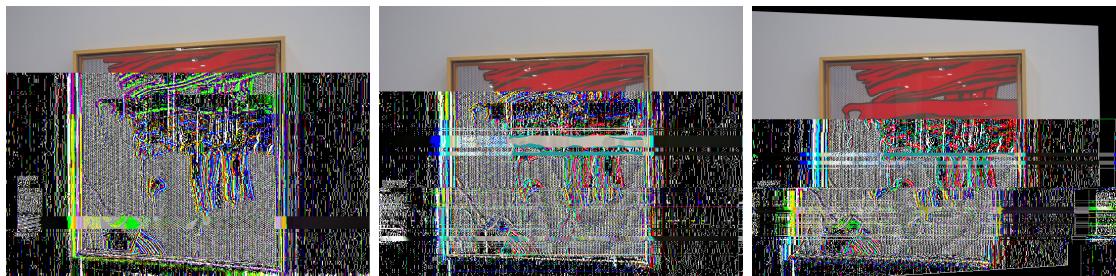


Figure 4: A gauche : image de travail, au centre : image de référence, à droite : image de travail après recalage par rapport à l'image de référence. Noter la différence de position des reflets entre l'image de référence et l'image recalée

2.1.3 Amélioration

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 est suffisante pour la pluspart des améliorations, mais n'est pas parfaites, en effet, les déformations dues à la lentille de la caméra induise des imperfections dans les images. Une simple perspective ne peut pas corriger ces déformations, et le recalage n'est pas idéal.

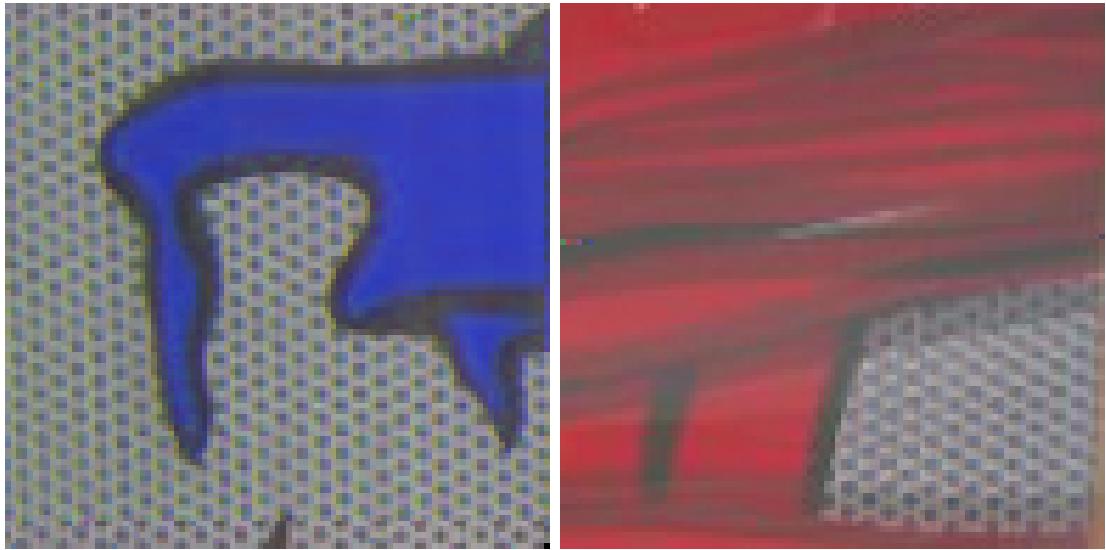


Figure 5: Détails de la moyenne de l'image recalées et de l'image de référence. A gauche, le recalage est correcte et les contours sont nets. A droite, sur le bord de l'image, le recalage est moins bon, et les contours sont flous.

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 abberants, inutilisable pour la fusion d'image.

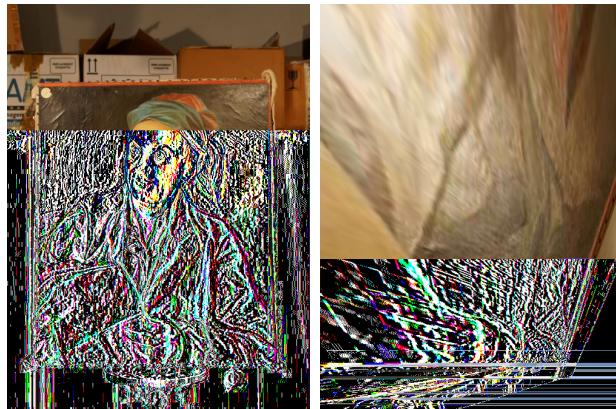


Figure 6: Image de référence, et image incorrectement recalés

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.

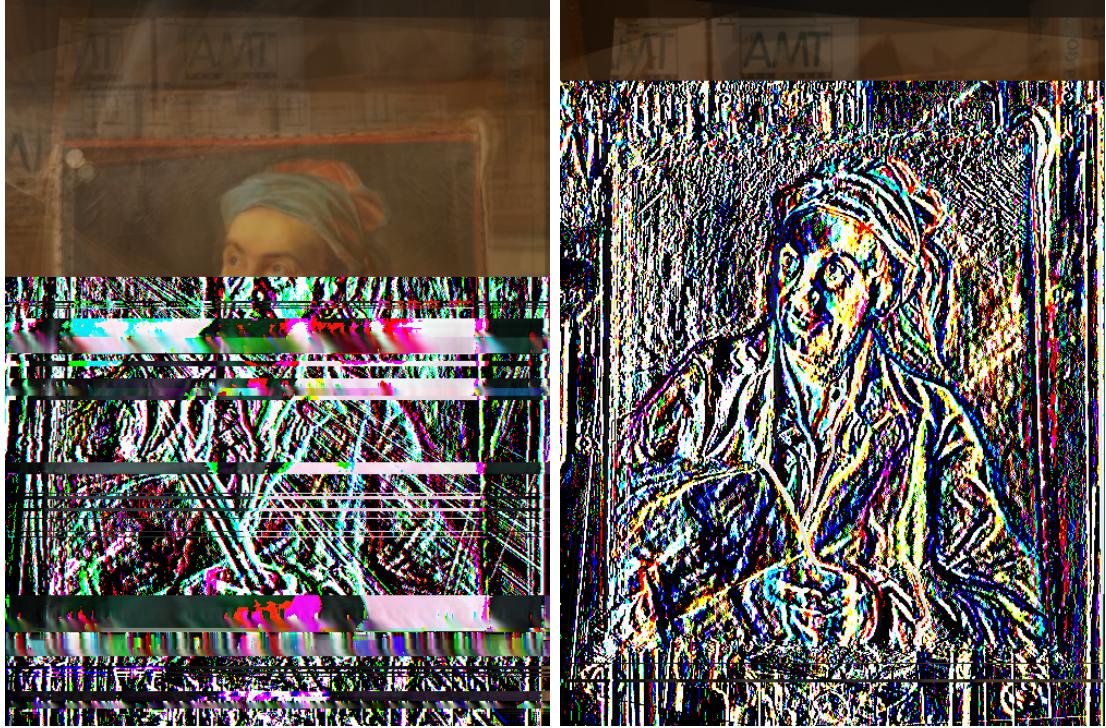


Figure 7: Moyennes des images recalées sans rejet (toutes les images recalées sont gardées) et avec rejet (les images où le nombre d'inlier est trop faible est écarté)

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

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

De nombreux choix sont possibles pour la fonction f . 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 par une méthode légèrement différente dans un autre espace de couleur, détaillée 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.

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.

2.2.2 Fusion par médiane car canal

2.2.3 Fusion par médiane vectorielle

Une autre façon consiste à voter, pixel par pixel, pour ce qui devrait être le pixel sans reflet. On suppose ici que le reflet se trouve dans moins de la moitié des images à un endroit donné. Deux méthodes sont alors possibles. On peut prendre la médiane des pixels, canaux de couleurs par canaux de couleurs. Cette méthode, simple et intuitive, a le désavantage, comme la méthode précédente, 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 ; il s'agit du vecteur minimisant la somme de ses distances L_1 aux autres vecteurs. En pratique, nous n'observons pas de différences entre ces deux médianes.

2.2.4 Fusion par médiane dans l'espace Lab

2.2.5 Fusion dans l'espace gradient

Enfin, une méthode décrite dans [?] permettant de réduire plus les reflets consiste à prendre la médiane vectorielle dans l'espace gradient des images. Pour ce faire, nous calculons les images gradients des images recalées, converties en niveau de gris, et nous en calculons la médiane vectorielle en chaque point. Ensuite, nous construisons les gradients de chaque canal de couleur de l'image fusionnée en prenant, pour chaque point et chaque canal, le gradient de l'image dont la médiane est issue. Ce sont donc bien les gradients des canaux de couleurs qui sont considérés.

Lorsque l'on connaît le gradient F d'une image, il est possible de la retrouver en minimisant $\min \|\text{grad}I - F\|$. Ceci se fait grâce à l'équation de Poisson $\Delta I = \text{div}F$. Cette équation est alors résolue, canal par canal, en passant par la transformée de Fourier, s'inspirant de [?]. Rencontrant des difficultés dans cette dernière étape, nous n'avons pas pu terminer cette méthode. Elle donne cependant d'excellents résultats dans [?], éliminant certains reflets apparaissant dans une majorité d'images.

3 Implémentation

3.1 Choix technique

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

Figure 8: Correction des artéfacts. De haut en bas : image de référence, puis corrigée avec les méthodes du minimum et de la médiane.

3.2 Architecture

La méthode décrite 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 médiante. 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 différente 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

Les résultats varient peu selon les méthodes (figure 8). Les médianes permettent de retirer de nombreux artéfacts de reflets, mais ne fonctionnent pas là où une majorité d'images présentent des artéfacts. Les méthodes de médiane, canal par canal ou globale, donnent les mêmes résultats sur les exemples que nous avons essayés.

5 Améliorations possibles

Plusieurs amélioration 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 contrastes 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).