

Spécification d'histogrammes dans le cas de zones constantes

Pierre Dubreuil, Thomas Eboli

March 5, 2017

Abstract

Dans ce rapport, nous allons présenter nos résultats au sujet d'application de la spécification d'histogrammes au traitement des images. Nous nous sommes appuyé sur le code fourni par madame Nikolova ainsi que sur le papier de base. La première étape du travail, en amont de tout bout de code a été de comprendre les enjeux soulevés par le papier et d'appréhender la puissance du speed-up de la phase "d'ordering" dans le problème de la spécification d'histogrammes. L'un des principaux problèmes que nous avons cherché à résoudre est l'a présence de "large pixels" dans les images obtenues après spécification. Nos méthodes de résolution reposent sur des considérations probabilistes sur les histogrammes, comme la forme de la distribution de probabilité d'une variable aléatoire X qui est ici l'image.

Ce document est organisé comme suit: la partie 1 est dédiée à présenter le problème de "large pixels" et d'en comprendre la source. La partie 2 présente nos premières approches pour résoudre le problème ainsi que que les résultats obtenus. La partie 3 s'intéressera à la méthode la plus satisfaisante de notre point de vue. IL s'agit de combiner la détexttion de pics dans l'histogramme avec des techniques dites "TMR".

1 Présentation du problème de "large pixels"

Lors de l'application de l'algorithme donné, on s'aperçoit que les images font apparaître des détails que l'on ne veut pas. La figure 1 illustre à merveille ce phénomène qui découle de la mauvaise quantification de la palette de couleurs dans la nouvelle configurations de bits de couleurs. En effet les figures 2 et 3 montrent que la spécification d'histogramme, lorsque l'histogramme cible est très différent de l'original, déplace des pixels d'un certain niveau de couleur vers des régions où il n'y avait pas de pixels pour colorer avec cette teinte. Le résultat est visible sur ces figures; le contraste est beaucoup plus riche et la photographie du port de Malte est moins terne mais on fait apparaître dans le ciel des blocs de pixels blancs. Ces pixels sont le résultat d'une volonté de rendre plus contrastée cette zone de l'image mais l'image originale ne possédant qu'une zone quasi constante de couleurs, la nouvelle allocation arrive à rendre le dégradé souhaité que par de gros tas de pixels, les "large pixels". Il faut donc arriver à trouver une technique permettant de gommer ces détails liés à des zones planes dans les images de départ.



Figure 1: Illustration de l'apparition de pixels nuisibles dûs à la ré-allocation de bits dans l'histogramme produisant une mauvaise quantification des couleurs.



Figure 2: Image de départ et son équivalent après spécification. On remarque qu'en haut à gauche, il y a des pixels apparents dans le ciel dûs à une mauvaise allocation des couleurs après spécification d'histogramme.

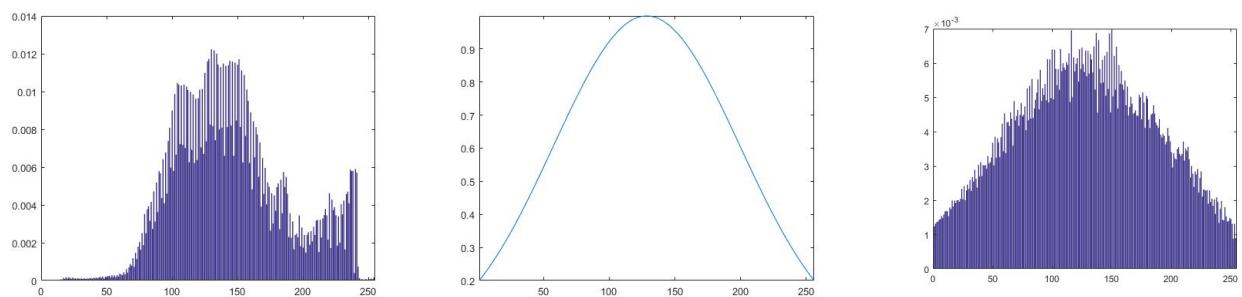


Figure 3: Gauche: histogramme de départ, Milieu: forme de l'histogramme cible, Droite: histogramme obtenu après application de la spécification par la routine order

2 Divers pistes

Dans cette section, nous souhaitons faire part des différentes approches que nous avons entrepris pour résoudre le problème de "large pixel".

2.1 Approche par tilling

Une première idée a été de vouloir traiter des bouts d'images indépendamment les uns des autres car le problème est souvent localisé dans certaines zones de l'image. Les approches par tilling et pyramides multi-échelles étant monnaie courante en traitement des images, nous avons voulu l'essayer dans notre problème. Malheureusement, qui dit tilling dit traitement séparé des informations ce qui ne nous arrange pas car nous voulons dispatcher l'ensemble des pixels de l'image et les réorganiser. Le résultat obtenu (sans utiliser d'algorithme type Midway qui aurait permis d'unifier les couleurs des différentes imagettes) n'est pas convaincant du tout car des zones claires et foncées apparaissent là où on le veut pas comme on peut le voir dans la figure 4 où les jonctions de chacune des imagettes pose problème. Cette figure a été obtenue avec une approche de type pyramide où chaque quart est la moyenne entre le quart de l'image de niveau 0 et l'imagette traitée à part au niveau 1 avec son propre histogramme (cf code dans le dossier "tiling").

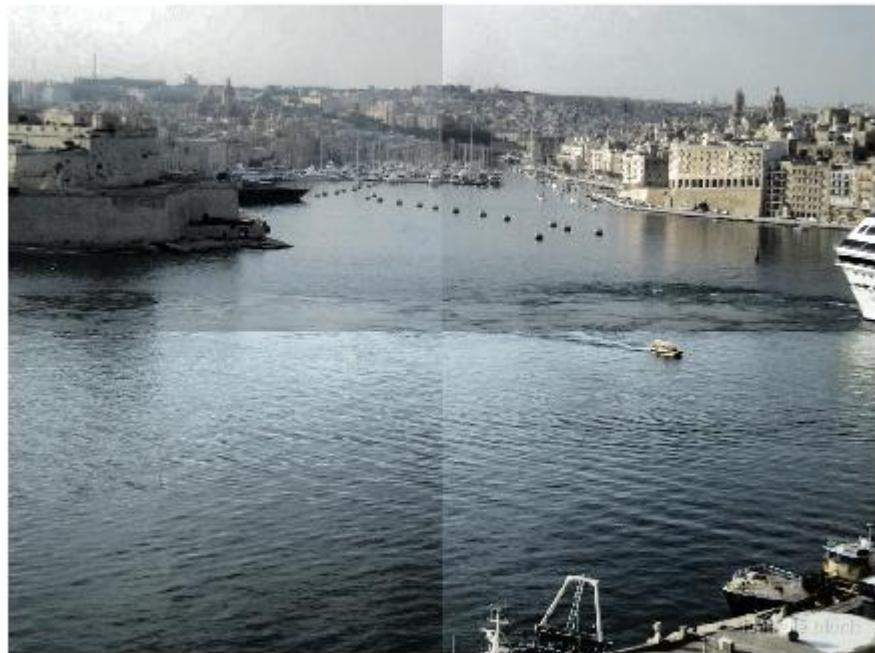


Figure 4: Sans même se donner la peine de traiter l'image par un algorithme type midway pour unifier les couleurs, on aperçoit dans l'image en haut à droite que le ciel est un dégradé qui ne prend pas en compte le ciel de l'imagette en haut à gauche. Pire encore, le phénomène de "large pixels" est encore plus accentué car on a encore moins de bits à allouer aux zones claires ! Même constat pour chacune des jonctions.

2.2 Résolution par ajout de bruit

Approche probabiliste

La partie précédente met en évidence que le transport d'un histogramme d'une image réelle vers un histogramme cible ne peut se faire ni directement ni par découpages successives. On obtient toujours des défauts dans l'image générée à partir du nouvel histogramme.

Suivant nos idées initiales, nous sommes partis du constat qu'une image est la réalisation d'une variable aléatoire X vivant dans un espace de très grande dimension (de l'ordre du million en pratique) et que son ou ses histogrammes, selon que X est une image couleur ou en niveau de gris, représentent les distributions de probabilités de chacun des canaux. Dans ce contexte, on peut utiliser un résultat classique de probabilités:

Théorème 1: Soient x et y les réalisations de deux variables aléatoires X et Y de densités respectives f_x et f_y . Alors $x + y$ est aussi une variable aléatoire suivant la loi $X + Y$ et de densité $f_x * f_y$.

Toutes les considérations suivantes seront prises sur des images en niveaux de gris. Pour appliquer les résultats à des images couleurs, il suffit d'appliquer les calculs à chacun des canaux. On considérant maintenant que x est notre image et en notant b le bruit (gaussien, uniforme, peu importe), on obtient que l'image $x_b = x + b$ possède un histogramme qui s'écrit $h_x * h_b$. On peut donc maintenant jouer sur le type de bruit et son écart-type σ pour modifier l'allure de l'histogramme de x et donc modifier en aval le contraste de manière plus fine.

Théoriquement, on arrive alors à moyenner l'histogramme de x . On définit la valeur d'un bin $h_{x_b}(i)$ pour $i \in 0, \dots, 255$ comme une fonction de $h_x(i)$ mais aussi de ses voisins. On force donc l'histogramme à avoir des valeurs faibles aux endroits où il n'y avait pas de pixels au départ et où l'algorithme faisait apparaître beaucoup de pixels. Cela réduit l'apparition de problèmes de larges pixels. Le choix du type de bruit à appliquer définit aussi le type de moyenne que l'on utilise. Par exemple un bruit uniforme appliquera une moyenne classique aux bins de l'histogramme de x .

2.3 Méthodes de dithering

L'idée d'ajouter du bruit à l'image pour casser ces zones constantes n'est pas nouvelle et date du milieu des années 50. Entre temps, le domaine de la quantification des palettes de couleurs a connu un grand intérêt avec l'arrivée des écrans couleurs. La trame de Bayer est l'un des exemples les plus connus, permettant de coloriser par des hachures des images. Globalement cette technique fonctionne bien mais fait apparaître des artefacts dans les zones hautes-fréquences. Notre problème étant celui de traiter de coloriser des zones de teinte constante qui deviennent à teinte variable après application de l'algorithme. La littérature à ce sujet est assez riche et plusieurs algorithmes sont disponibles comme celui de Floyd et Steinberg de 1975. Nous avons choisi l'algorithme d'Atkinson datant des années 80 qui repose sur la technique de Floyd et Steinberg mais est plus rapide et donne de meilleurs résultats.

Le principe de ces algorithmes est assez simple. Pour chaque pixel codé, on attribut une valeur q_i appartenant aux valeurs de quantifications possibles $q_j, \forall j \in 1, \dots, L$ puis on garde en mémoire l'erreur entre cette valeur quantifiée q_i et la véritable valeur de ce pixel. On la propage aux pixels voisins et on quantifie la valeur voisine en ajoutant l'erreur de quantification du premier pixel ce qui permet de créer des structures en damier ou des sortes de patchworks plus ou moins denses de pixels noirs et blancs pour donner l'impressions de plusieurs teintes de gris alors qu'on a que deux couleurs. Dans le cas des images couleurs, le procédé est appliqué à chacun des canaux de couleurs.

Avec cette procédure simple mais astucieuse, on arrive à gommer beaucoup des erreurs de quantifications. L'idée du procédé n'est pas étrangère à notre première approche du problème car même si nous avons pensé en terme d'histogramme au début, ajouter du bruit revient à casser la redondance des zones constantes et donc à donner un aspect plus contraster et moins bloc de couleurs dans les zones unis, les zones hautes-fréquences ne faisant pas ressortir le grain du bruit.

2.4 Résolution par étude de la fonction de répartition

Une approche plus en lien avec le papier expliquant la construction de la méthode de tri et d'ordonnancement est de s'intéresser à la fonction de répartition de l'image. Le code de base fournit une fonction pour obtenir un histogramme cible à partir d'une gaussienne. Dans le cas de l'image du bungalow 5, le résultat est très bon. En adaptant les bornes L et R de la fonction, on peut trouver de bons résultats pour des images dont la fonction de répartition est relativement simple (aucun point d'inflexion, allure d'une fonction de répartition d'une gaussienne, comportement très régulier...). La figure 7 illustre ce constat. Il suffit de voir le résultat sur l'image "Malta" 6 où on n'aperçoit plus du tout les "large pixels" une fois les bons paramètres L et R trouvés ($L = 0.2$, $R = 0.5$).



Figure 5: Cas où tout se passe bien, celui de l'image de bungalow.



Figure 6: A gauche, l'image de départ, au milieu, l'image obtenue avec une gaussienne répartissant mal les pixels et à droite image obtenue avec un choix de gaussienne plus judicieux qui distribue plus de pixels dans les zones clairs pour améliorer l'allure du dégradé.

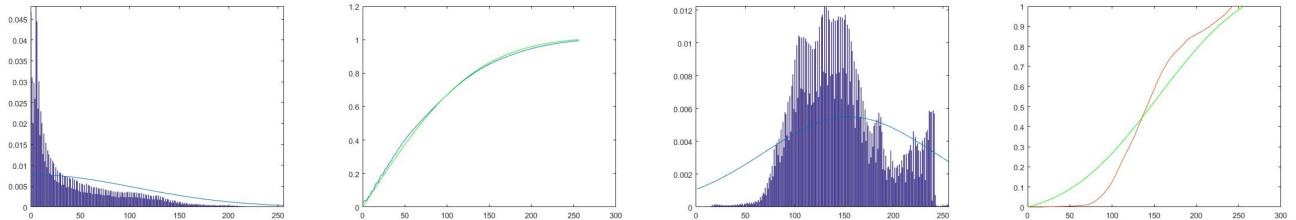


Figure 7: A gauche, histogrammes et histogrammes cumulés pour la photo du bungalow et à droite, la même chose pour la photo de Malte. Dans le cas du bungalow, on trouve un histogramme qui permet une spécification exacte comme expliqué dans l'article. Dans le cas de Malte, on trouve une fonction approchée, mais qui donne un résultat très satisfaisant.

Optimisation des paramètres

Le but est alors de trouver la fonction modélisant le plus finement l'histogramme de départ. Nous avons utilisé la routine hsGauss.m pour coller au mieux à l'histogramme de départ tout en distribuant des pixels sur toute l'amplitude de niveaux de gris allant de 0 à 255. Il faut donc jouer sur les paramètres R et L qui définissent la gaussienne cible. On tombe sur un problème de "curve fitting" qui peut s'écrire (h désigne l'histogramme de départ et g la gaussienne obtenue par hsGauss):

$$\min_{(L,R)} \|h - g(L,R)\|^2 \quad (1)$$

Comme L et R sont compris entre 0 et 1, on doit ajouter quatre contraintes inégalités au problème. Si on pose $u = (L, R)$, alors on a $h_1(u) = -L$, $h_2(u) = L - 1$, $h_3(u) = -R$ et $h_4(u) = R - 1$. Le problème s'écrit alors avec les λ_i des réels positifs:

$$\min_u \|h - g(u)\|^2 + \sum_{i=1}^4 \lambda_i h_i(u) \quad (2)$$

On a réussi à poser correctement le problème sous la forme d'un problème d'optimisation quadratique sous contraintes. Un tel problème peut se résoudre par la méthode d'Uzawa par exemple et le code utilisé se situe dans le dossier *optim_R_L* du code. Malheureusement comme nous le verrons dans l'exemple suivant, l'optimisation des paramètres ne suffit pas à effacer les problèmes de large pixels.

Limite de la méthode

En effet, Pour des images où la répartition du contraste est beaucoup plus localisé et que l'on a plusieurs clusters de niveaux de gris par exemple, la modélisation par une gaussienne dessinée à la main ou optimisée est trop grossière comme le montre la figure 8. On remarque sur l'histogramme cumulé la présence de deux paliers atteints qui correspondent à la présence de deux pics dans l'histogramme. Dans ces conditions, une approximation gaussienne ne permet pas d'imiter ce comportement. C'est le cas pour l'image des oiseaux par exemple. Le fond étant presque uni et clair, un histogramme cible gaussien va avoir tendance à redistribuer trop largement les pixels et faire apparaître des dégradés sur l'image synthétique, c'est un problème analogue aux larges pixels.

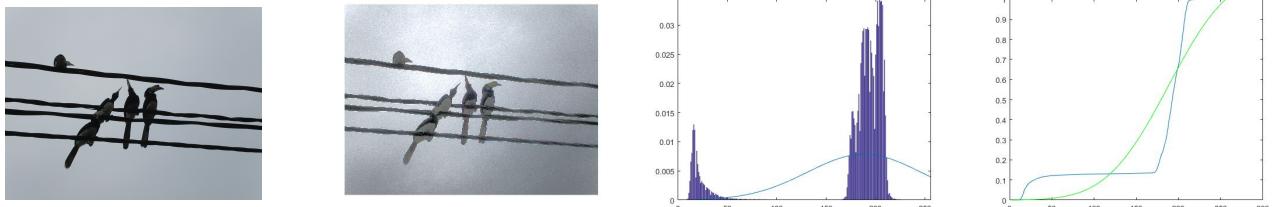


Figure 8: De gauche à droite: l'image originale, l'image "améliorée", l'histogramme de l'image de départ et la fonction cible et les fonctions de répartition associées. On remarque que le bord supérieur droit est moins sombre dans la dernière image, conséquence de l'allocation des pixels en faveur des zones claires.

Une idée supplémentaire serait d'avoir une fonction pas trop régulière car celle-ci risquerait d'allouer trop de pixels à des bins non présents sur l'image d'origine. On peut alors imaginer trouver des fonctions plus compliquées que la gaussienne qui régularisent l'histogramme cumulé et qui s'adaptent aux problèmes de sauts brusques dans la fonction de répartition. Cependant, même en trouvant de telles fonctions, d'autres histogrammes cibles poseront sûrement problème et il faudrait alors trouver un autre type de fonction qui approximerait l'histogramme cible.

3 Suppression des larges pixel par TMR

Nous avons vu que par l'approximation de l'histogramme par une gaussienne puis l'application de l'algorithme de *ranking* nous obtenons deux types de résultats. Soit le résultat est satisfaisant car l'histogramme de base peut facilement être approximé par une gaussienne, soit la forme de l'histogramme est très éloigné d'une gaussienne et la redistribution des pixels fait apparaître (le plus souvent dans les zones d'aplats de couleur) le problème des larges pixels. Dans cette partie nous présentons une solution qui combine deux approches: approximation de l'histogramme par une somme de gaussiennes puis nous appliquons un algorithme TMR tel que définit dans [1] pour faire disparaître les large pixels. On rappelle ici l'algorithme en question qui permet de réadapter les contrastes d'une image spécifiée pour que les artefacts de celle-ci disparaissent.

$$TMR_u[(g(u)] = u + Y_u[g(u) - u] = Y_u[g(u)] + u - Y_u[u] \quad (3)$$

Où u est l'image de départ, $g(u)$ est l'image spécifiée et $Y_u[w](x) = \frac{1}{C(x)} \sum_{y \in N(x)} w(y) e^{\frac{\|u(x) - u(y)\|^2}{\sigma^2}} dy$ est le filtre guidé. On répète N fois cette opération pour

3.1 Détection de pics et choix de la forme de l'histogramme cible

3.1.1 Détection des pics

En se basant sur le code *hsGauss.m* nous avons écrit une fonction qui permet de repérer les pics sur l'histogramme et de fitter la meilleure gaussienne dessus. Dans un deuxième temps nous avons rajouté un calcul de l'écart type autour de ce pic pour choisir entre fitter une gaussienne ou une laplacienne qui correspond à un pic plus étroit adaptant encore plus l'histogramme cible à celui de départ mais les résultats ne sont pas plus concluants qu'avec

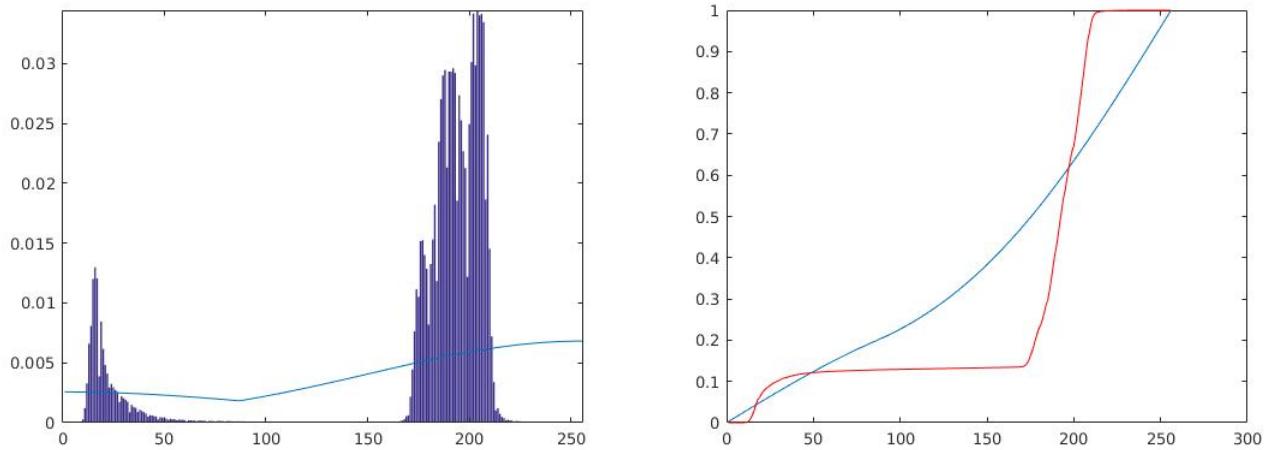


Figure 9: Notre algorithme a reconnu deux pics et dessine l'histogramme cible comme étant le maximum point par point des deux gaussiennes fittées sur chaque pic. En rouge, l'histogramme cumulé de départ et en bleu notre nouvel histogramme.

un de choix de gaussiennes et il est préférable de sélectionner la forme de cette dernière pour conserver l'aspect "étalement des bits" sur tout l'histogramme. L'histogramme que nous obtenons en sortie est le maximum point par point des gaussiennes fitter sur chaque pic, pondéré en fonction de la hauteur de celui-ci. Nous avons aussi choisi d'adapter les paramètres L et R des scripts *hsGauss.m* et *hsLaplace.m* en fonction de la position du maximum local détecté. Ainsi nous avons choisi de subdiviser l'espace des valeurs de pixels en 5 segments de longueur à peu près égales. En pratique nous avons adapter des paramètres L et R selon que le bon se situe dans [0, 50], (50, 100], (100, 150], (150, 200] et (200, 255] ce qui permet de coller grossièrement à l'histogramme de départ. Ceci rejoint notre tentative précédente d'optimiser sur L et R en résolvant un problème de curve-fitting. Cette simplification de la recherche L et R est un gain de temps énorme, une simplification de l'algorithme et en pratique donne d'excellents résultats.

Comme nous pouvons le voir dans la figure 9, nous avons rajouté un paramètre dans notre fonction *choosehs-Gauss.m* pour que deux pics trop rapprochés ne s'ajoutent pas dans le modèle. Ainsi, nous considérons qu'un bin est un pic lorsque celui ci est éloigné au moins de 50 bins de tous les pics précédemment repérés. C'est un choix totalement arbitraire mais qui en pratique fonctionne bien. Le résultat de notre méthode pour l'image bird dont l'histogramme de la figure 9 en est la répartition des pixels se trouve en figure 10.

Dans l'algorithme TMR, le choix d'un paramètre σ détermine la taille de la fenêtre du patch qu'on doit restaurer. Pour des choix autour de 1, notre algorithme réagit très bien. La figure ?? monter que pour plusieurs choix de ce paramètre autour de 1, des résultats plus ou moins satisfaisants sont visibles.

Limitations de notre méthode

Comme on peut le voir sur certaines images en Annexe A, notre méthode fonctionne globalement bien sur les images de test mais pour certaines, le changement de contraste n'est pas flagrant. L'algorithme de TMR semble détruire le changement de contraste en ramenant l'image modifiée vers l'image de départ.

Conclusion

Dans ce projet, nous avons donc cherché à résoudre le problème des larges pixels en jouant très fortement sur l'histogramme.



Figure 10: A gauche, l'image modifiée par l'algorithme de base, A droite, sa version corrigée par notre algorithme mêlant curve fitting et TMR. On remarque bien que notre méthode gomme très bien les défauts visibles à gauche tout en respectant le ton neutre du ciel.



Figure 11: De gauche à droite, σ vaut 0.8, 1 et 1.5. Le paramètre σ est primordial pour obtenir un bon résultat.

Notre première idée d’ajouter du bruit sur l’image s’est révélée être un bon point de départ. Fitter l’histogramme avec la meilleure gaussienne a permis de trouver qu’il fallait jouer sur le nombre de pics présents pour adapter au mieux le code donné. Enfin l’application de l’algorithme TMR à l’image obtenue par spécification de notre histogramme permet d’obtenir globalement des résultats très convaincants.

References

- [1] Julien Rabin, Julie Delon, and Yann Gousseau. Artefact-free color and contrast modification. working paper or preprint, July 2010.

Annexe A: Quelques exemples de notre méthode



Figure 12: Original



Figure 13: Naïf



Figure 14: Le notre



Figure 15: Original



Figure 16: Naïf

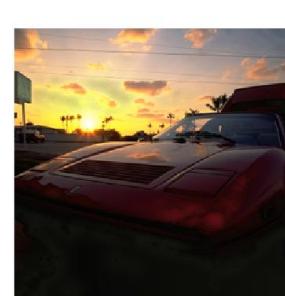


Figure 17: Le notre



Figure 18: Original



Figure 19: Naïf

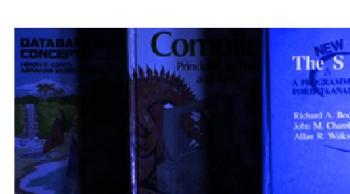


Figure 20: Le notre



Figure 21: Original



Figure 22: Naïf



Figure 23: Le notre

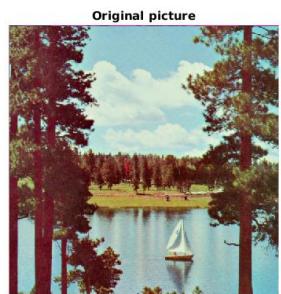


Figure 24: Original

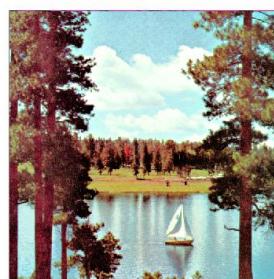


Figure 25: Naïf

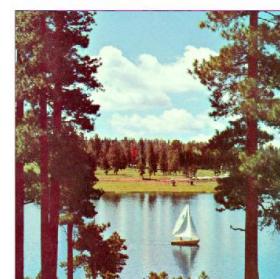


Figure 26: Le notre