

Transfert de couleur par transport optimal

DUSSERE Laurie, DUZ Elisa, GENDREAU Alexandre, JULIA Nicolas (Groupe 1)
NGUYEN Minh Hai, LE Cam Thanh Ha, CAPITAINE Amandine, AZEVEDO Tess (Groupe 2)

Janvier 2023

1 Présentation du problème

1.1 Introduction à Transport Optimal

Étant donné deux mesures de probabilité $\mu \in X$ et $\nu \in Y$, et une **fondation de coût** $c(\cdot, \cdot) : X \times Y \rightarrow \mathbb{R}^+$, dans OT, on cherche un **plan de transport optimal** $T : X \rightarrow Y$ qui minimise la quantité suivante

$$\inf \left\{ \int_X c(x, T(x)) d\mu(x) \mid T_*(\mu) = \nu \right\}.$$

où pour tout ensemble mesurable $A \subset Y$, T_* est défini comme suit

$$T_*(\mu)(A) = \mu(T^{-1}(A))$$

En d'autres termes, la solution T **transporte** la distribution μ vers la distribution ν avec la distance minimale (par rapport à la fonction de coût c).

Cette définition peut être illustrée par le problème de Monge, daté de 1781. Considérons que μ est un tas de sable et ν des trous (avec les même volume ou masse), si l'on considère que le coût de déplacement d'un volume élémentaire de sable d'un point à l'autre est proportionnel à la distance euclidienne (fonction de coût l_2), alors le plan de transport optimal vous donne le moyen le plus efficace de déplacer tout le sable de μ à ν .

1.2 Lien avec Transfert de couleur

L'objectif du projet est de recoloriser une image X à partir des pixels d'une image Y en utilisant le transport optimal.

Le lien entre Transport Optimale et transfert de couleur fait suite à l'observation que les histogrammes de couleur sont des mesures discrètes de Radon. En fait, une image elle-même n'est pas une distribution, mais on peut la représenter comme une **discrétisation d'un nuage de points**. Soit $f \in \mathbb{R}^{N \times d}$ une image de N pixels et d couleurs ($d = 3$ pour les images couleur, $d = 1$ pour les images en noire-blanche), chaque pixel peut être vu comme un point dans \mathbb{R}^d , notons $X = \{X_i\}_{i=0,1,\dots,N-1}$ (par exemple $X_i = (R_i, G_i, B_i)$ si $d = 3$) l'ensemble de tous les pixels, la distribution correspondante du nuage de points est la suivante

$$\mu_X = \frac{1}{N} \sum_{i=0}^{N-1} \delta_{X_i}$$

où δ_{X_i} est la distribution de Dirac à location X_i .

Considérons une image d'entrée u avec son nuage de points correspondant X et une image cible v avec son nuage de points correspondant Y (de même taille), on peut **transfert** les couleurs de v en u en résolvant un problème de transport optimal entre μ_X et μ_Y . Dans toute la suite, par abuse de notation, on va noter X, Y, μ_X, μ_Y pour l'image entrée, l'image cible et ses distributions, respectivement.


 (a) Image X (sur laquelle on va changer les couleurs)

 (b) Image Y (source des couleurs)

FIGURE 1 – Example d'images qu'on va utiliser.

2 Résolution du problème grâce à l'algorithme de Sinkhorn (Groupe 1)

Notons c la fonction de coût. La distance entre deux pixels $X_i = (R_i^X, G_i^X, B_i^X)$ et $Y_j = (R_j^Y, G_j^Y, B_j^Y)$ (des images couleurs) se calcule par la distance sur chacun des canaux de l'image :

$$c(X_i, Y_j) = \|X_i - Y_j\|^2 = (R_i^X - R_j^Y)^2 + (G_i^X - G_j^Y)^2 + (B_i^X - B_j^Y)^2$$

Nous noterons $C = (c_{X_i, Y_j})$, la matrice des coûts, ainsi que $P = (p_{i,j})$, la matrice des distributions des pixels des deux images. On peut donc traduire le problème du transport optimal des pixels de l'image Y vers l'image X par :

$$\underset{P \in U(\mu_X, \mu_Y)}{\operatorname{argmin}} \sum_{i,j} C_{i,j} P_{i,j} - \varepsilon H(P)$$

où :

- $U(\mu_X, \mu_Y) = \{P, P_{ij} \geq 0, \sum_j P_{ij} = a_i, \sum_i P_{ij} = b_j\}$ désigne l'ensemble des combinaisons possibles entre les pixels.
- $H(P) = -\sum_{ij} P_{ij} (\log(P_{ij}) - 1)$ correspond à l'entropie de la matrice P .
- ε correspond à la compromis entre les deux termes : minimiser le coût ou bien encourager une distribution homogène.

Enfin, en posant $K_{ij}^\varepsilon = \exp(-\frac{1}{\varepsilon} C_{ij})$, le problème devient :

$$\underset{P \in U(\mu_X, \mu_Y)}{\operatorname{argmin}} \sum_{i,j} P_{i,j} \log \left(\frac{P_{i,j}}{K_{i,j}^\varepsilon} \right)$$

Afin de résoudre ce problème d'optimisation, nous avons utilisé l'algorithme de **Sinkhorn** qui est un algorithme de projection alternée.

Cet algorithme résout le problème d'optimisation défini dans la section 1. Afin de l'appliquer, nous avons dû réduire la taille de l'image en regroupant les pixels par blocs. Vous pouvez trouver en annexe les résultats de transfert de couleurs obtenus ainsi qu'une représentation des pixels dans un espace en 3D (voir annexe A).

Un problème de cet algorithme est qu'il ne fonctionne pas pour des images de grandes tailles. En effet, il est nécessaire de calculer la matrice de coût C , ce calcul est extrêmement long pour des images en grandes dimensions. Ici, les images utilisées étaient de taille "raisonnable", le calcul s'effectuait donc en un temps fini.

Pour palier à ce problème, il existe des méthodes pour implémenter l'algorithme de Sinkhorn de façon optimisé en utilisant des convolutions. Nous n'avons pas implémenté ces méthodes mais simplement utilisé des algorithmes déjà codés pour observer le résultat.

Algorithm 1 Algorithme de Sinkhorn

```

 $u_0 \leftarrow \mathbb{1}_n$ 
 $v_0 \leftarrow \mathbb{1}_n$ 
for  $k$  in  $n\_iter$  do
     $u_{k+1} \leftarrow \frac{a}{Kv_k}$ 
     $v_{k+1} \leftarrow \frac{b}{K^T u_k}$ 
     $P \leftarrow \text{diag}(u) K \text{diag}$ 
end for

```

3 Résolution du problème grâce à la distance de Wasserstein (Groupe 2)

La fonction coût du transport optimal définit une métrique entre mesures de Radon. On peut définir la métrique de Wasserstein p^{ieme} , $\mathcal{W}_p(\mu, \nu)$ entre deux distribution μ, ν comme :

$$\mathcal{W}_p(\mu, \nu) := \left(\inf \left\{ \int_X c(x, T(x))^p d\mu(x) \mid T_*(\mu) = \nu \right\} \right)^{1/p}$$

La **distance quadratique de Wasserstein** entre deux nuages de points X et Y de même taille N est donc définie par

$$W(\mu_X, \mu_Y)^2 = \min_{\sigma} W_{\sigma}(\mu_X, \mu_Y) \quad \text{où} \quad W_{\sigma}(\mu_X, \mu_Y) = \sum_{i=0}^{N-1} \|X_i - Y_{\sigma(i)}\|^2 \quad (1)$$

où σ est une permutation de N éléments qui définit une correspondance entre les deux nuages de points. Pour simplifier, dans ce qui suit, $W(\mu_X, \mu_Y)$ sera désigné par $W(X, Y)$.

En dimension 1, l'affectation σ qui minimise le coût précédent est celle qui préserve l'ordre des points et l'affectation optimale peut être calculée en $O(N \log(N))$ grâce à un algorithme de tri rapide. Trouver la solution de ce problème à l'aide de la distance de Wasserstein dans une dimension supérieure à 1 est trop exigeant en calcul et donc pas applicable au traitement d'image. On considère donc une métrique alternative, nommée **Sliced Wasserstein Distance** et définie par

$$\begin{aligned} SW(X, Y)^2 &= \int_{\theta \in \Omega} W(X_{\theta}, Y_{\theta})^2 d\theta \quad \text{où} \quad X_{\theta} = \{\langle X_i, \theta \rangle\}_{i=0, \dots, N-1}, Y_{\theta} = \{\langle Y_i, \theta \rangle\}_{i=0, \dots, N-1} \\ &= \int_{\Omega} \left(\sum_{i=0}^{N-1} |\langle X_{\sigma_{X, \theta}(i)} - Y_{\sigma_{Y, \theta}(i)}, \theta \rangle|^2 \right) d\theta \end{aligned}$$

avec $\Omega = \{\theta \in \mathbf{R}^d : \|\theta\| = 1\}$ la sphère unité.

De nouveau, le calcul de l'affectation optimale σ^* dans ce cas est toujours insoluble. Par conséquent, au lieu de calculer la solution explicite, nous utilisons un **algorithme de descente de gradient stochastique** qui calcule la solution d'une approximation du problème de transport optimal sur des distributions 1D.

Remarquons que nous voulons faire correspondre le nuage de points X à Y , donc si nous fixons Y , la fonction $E_Y(X) \stackrel{\text{def.}}{=} SW(X, Y)$ est lisse et son gradient (par rapport à X) est un vecteur dont la i -ième composante est

$$\nabla E_Y(X)_i = 2 \int_{\Omega} (\langle X_i - Y_{\sigma_{Y, \theta} \circ \sigma_{X, \theta}^{-1}(i)}, \theta \rangle \theta) d\theta$$

En utilisant un pas de descent suffisamment petite τ , l'itération de descente de gradient stochastique s'écrit comme suit

$$X \leftarrow X - \tau \nabla E_Y(X)$$

Ce gradient peut être calculé en remplaçant l'intégrale par une somme finie contre des directions tirées au hasard $\theta \in \Omega$. Mais numériquement, au lieu de calculer le gradient approché par une estimation de Monte Carlo,

Algorithm 2 Algorithme de minimisation Sliced Wasserstein Distance

Require: (X, Y, τ, n_iter) où

- X nuage de points d'entrée
- Y nuage de points visé
- τ le pas de descente (0.1 par exemple)
- n_iter : nombre d'itérations
- Initialiser Z image de sortie : $Z \leftarrow X$
- for** k in n_iter **do**

 - Choisir une direction aléatoirement $\theta \in \Omega$
 - Projection de Y et Z sur θ : $Y_\theta = \{\langle Y_i, \theta \rangle\}_{i=0, \dots, N-1}$ et $Z_\theta = \{\langle Z_i, \theta \rangle\}_{i=0, \dots, N-1}$
 - Permuter les points projetés par Transport 1D : $\sigma_Y, \sigma_Z = \text{Transport1D}(Y_\theta, Z_\theta)$ (voir Algorithm 3)
 - Mise à jour par descente de gradient stochastique $Z[\sigma_Z] \leftarrow Z[\sigma_Z] - \tau (\langle Z_\theta[\sigma_Z] - Y_\theta[\sigma_Y], \theta \rangle \theta)$

- end for**

nous pouvons décomposer le gradient à chaque itération avec une direction aléatoire θ . Voici l'algorithme final utilisé :

La permutation à chaque itération peut être calculée grâce à l'algorithme Transport 1D défini ci-dessous.

Algorithm 3 Transport 1D

Require: X, Y two point clouds

- $\sigma_X \leftarrow \text{argsort}(X)$
- $\sigma_Y \leftarrow \text{argsort}(Y)$
- return** σ_X, σ_Y

4 Post-processing

Un inconvénient courant de ce transfert de couleur utilisant le transport optimal est la révélation d'artefacts, tels que le mélange des couleurs, l'atténuation des détails et des textures.

Notons $T(X)$ l'image après transfert de couleur, dans cette partie nous appliquons une méthode de filtrage générique permettant de supprimer les artefacts tout en préservant les détails (voir cet article pour plus de détails). On observe que

$$T(X) = X + T(X) - X$$

Le principe est d'appliquer une modification de filtrage à la **différence**, notée $M(X) = T(X) - X$. Ainsi, pour réduire les artefacts observés dans $T(X)$, on peut filtrer $M(X)$ par un filtre (ou opérateur) H et ensuite l'image finale peut être obtenue par

$$X + H(M(X)) = X + H(T(X) - X)$$

4.1 Lissage de l'image par un filtre bilatéral (Groupe 1)

Nous avons appliqué un filtre bilatéral sur la différence entre l'image originale et l'image que nous avons obtenu grâce à l'algorithme de Sinkhorn (c'est-à-dire l'image avec les nouvelles couleurs). Ce filtre bilatéral permet d'obtenir un résultat plus lisse. En effet, afin d'appliquer l'algorithme de Sinkhorn, nous avons dû réduire la taille de l'image, en regroupant des pixels par blocs. Ainsi, les résultats ne sont pas forcément très "propres" en termes de transfert de pixels.

L'algorithme est appliqué pixel par pixel. L'idée est d'appliquer une moyenne pondérée par les pixels autour d'un pixel d'intérêt. L'algorithme est donné ci-dessous :

- Définition de la fenêtre, selon un certain seuil de voisinage
- Calcul des poids relatifs au sens de la distance RGB
- Calcul de la moyenne pondérée sur chaque fenêtre et attribution de la valeur au pixel d'intérêt

Les résultats sont donnés en sous-section A.2.

Un problème du filtre bilatéral est que, comme l'algorithme de Sinkhorn, il n'est pas adapté pour les images de grandes tailles car il nécessite de traiter individuellement chaque pixel de l'image. Pour palier à ce problème, il existe notamment les filtres guidés, qui permettent d'avoir des résultats de lissage très satisfaisants et moins coûteux en temps. Les résultats pour le filtre guidé sont donnés en sous-section A.3.

4.2 Lissage de l'image par un filtre moyen (Groupe 2)

Un filtre simple qui fonctionne pas mal est le filtre moyen. Dans ce cas, l'opérateur H devient juste une convolution :

$$H : Z \mapsto h * Z$$

où h est un petit filtre de taille $m \times m$ rempli par $\frac{1}{m^2}$.

4.3 Lissage de l'image par un filtre guidé (Groupe 2)

Nous définissons d'abord un filtrage linéaire qui est variant par translation qui implique une image de guidage I , une image d'entrée p (à filtrer) et une image de sortie q . Les deux images I et p sont données à l'avance selon l'application, et elles peuvent être identiques.

La sortie du filtrage au niveau d'un pixel i est exprimée sous forme de moyenne pondérée :

$$q_i = \sum_j W_{ij}(I)p_j$$

où i et j sont des indices de pixel. Le noyau du filtre W_{ij} est une fonction de l'image de guidage I et indépendant de l'image d'entrée p . Ce filtre filtre est linéaire par rapport à p .

L'hypothèse clé du filtre guidé est un modèle linéaire local entre le guidage I et la sortie du filtrage q . Nous supposons que q est une transformée linéaire de I dans une fenêtre ω_k centrée sur le pixel k :

$$q_i = a_k I_i + b_k, \forall i \in \omega_k$$

où (a_k, b_k) sont des coefficients linéaires supposés constants dans ω_k . Nous utilisons une fenêtre carrée de rayon r . Ce modèle linéaire local garantit que q a un bord seulement si I a un bord car $\nabla q = a \nabla I$.

Pour déterminer a_k et b_k , nous cherchons à minimiser la différence entre q et p tout en maintenant la relation linéaire. Plus précisément, nous minimisons la fonction suivante dans la fenêtre ω_k .

$$E(a_k, b_k) = \sum_{i \in \omega_k} ((a_k I_i + b_k - p_i)^2 + \epsilon a_k^2)$$

où ϵ est un paramètre de régularisation pénalisant les grands a_k .

La solution est donnée par

$$a_k = \frac{\frac{1}{|\omega|} \sum_{i \in \omega_k} I_i p_i - \mu_k \bar{p}_k}{\sigma_k^2 + \epsilon}$$

et

$$b_k = \bar{p}_k - a_k \mu_k$$

où μ_k et σ_k sont la moyenne et variance de I dans ω_k , $|\omega|$ est le nombre de pixels dans ω_k et \bar{p}_k est la moyenne de p dans ω_k .

Dans cet article, ils ont proposé un algorithme pour appliquer le filtre guidé, avec \odot désigne le produit terme à terme, \oslash désigne la dévision terme à terme et f_{mean} désigne le filtrage par un filtre moyen comme dans sous-section 4.2.

Algorithm 4 Filtre guidé

Require: L'image d'entrée p , l'image de guidage I , rayon r , régularisation ϵ
Ensure: Image de sortie filtrée q

```

 $\text{mean}_I = f_{\text{mean}}(I)$ 
 $\text{mean}_p = f_{\text{mean}}(p)$ 
 $\text{corr}_I = f_{\text{mean}}(I \odot I)$ 
 $\text{corr}_{Ip} = f_{\text{mean}}(I \odot p)$ 
 $\text{var}_I = \text{corr}_I - \text{mean}_I \odot \text{mean}_I$ 
 $\text{cov}_{Ip} = \text{corr}_{Ip} - \text{mean}_I \odot \text{mean}_p$ 
 $a = \text{cov}_{Ip} \oslash (\text{var}_I + \epsilon)$ 
 $b = \text{mean}_p - a \odot \text{mean}_I$ 
 $\text{mean}_a = f_{\text{mean}}(a)$ 
 $\text{mean}_b = f_{\text{mean}}(b)$ 
return  $q = \text{mean}_a \odot I + \text{mean}_b$ 

```

A Sortie de l'algorithme de Sinkhorn

A.1 Sans régularisation

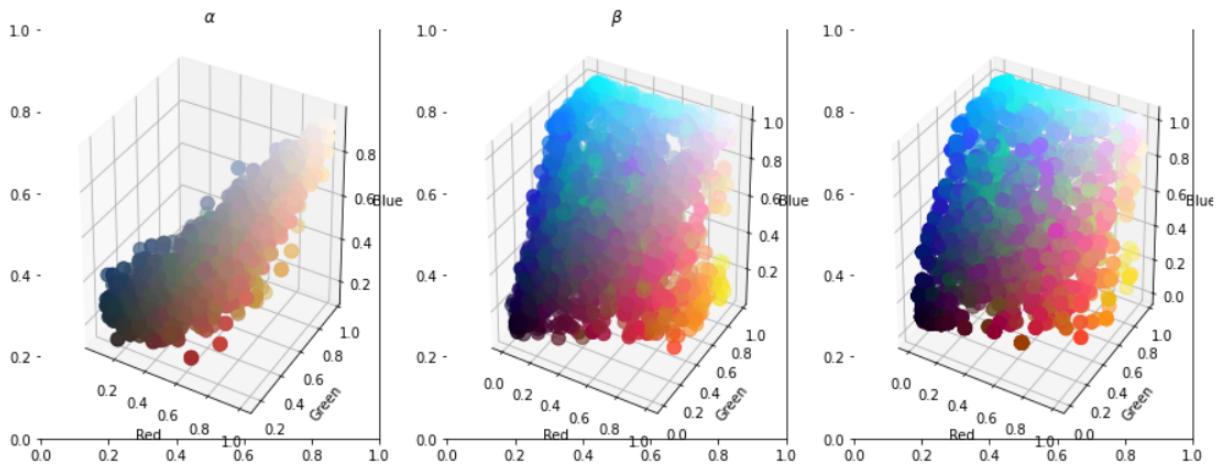
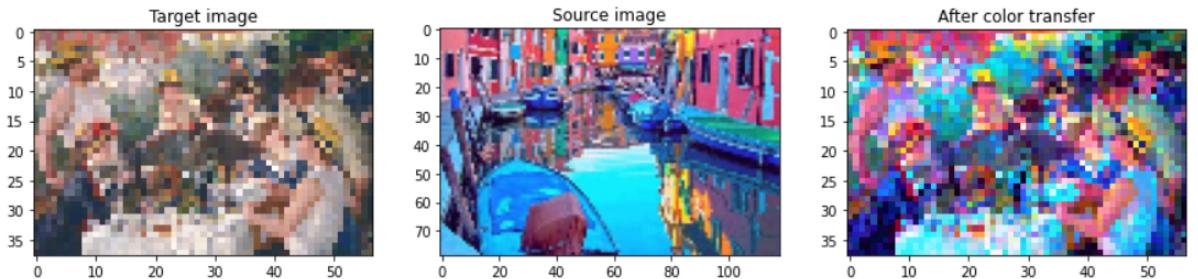


FIGURE 2 – Sortie de l'algorithme Sinkhorn. A gauche, image de départ, sur laquelle on veut modifier les pixels, et la distribution de ses pixels. Au milieu, l'image source avec la distribution de ses pixels. A droite, l'image modifiée, ainsi que sa distribution. On voit bien que l'algorithme a convergé et que la distribution de l'image finale est équivalente avec celle de l'image source.

A.2 Sortie de l'algorithme après lissage par filtre bilatéral

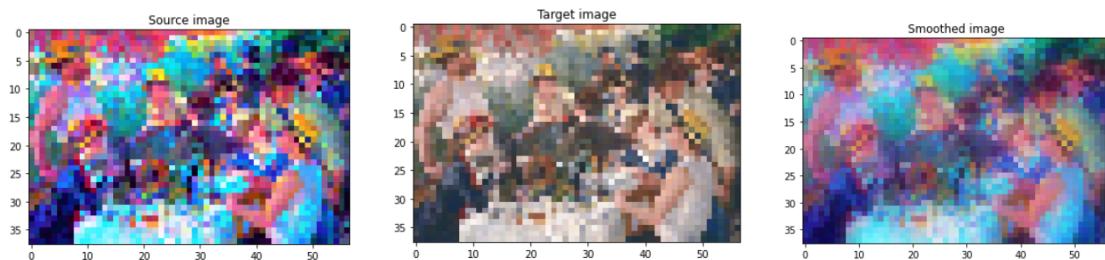


FIGURE 3 – A gauche, l'image obtenue après transfer de couleur grâce à l'algorithme de sinkhorn, au milieu l'image source et à droite l'image obtenue après lissage avec filtre bilatéral.

A.3 Sortie de l'algorithme après lissage par filtre guidé



FIGURE 4 – A gauche, l'image obtenue après transfert de couleurs grâce à l'algorithme de Sinkhorn optimisé, au milieu l'image source et à droite, l'image obtenue après lissage par filtre guidé.

B Sortie de l'algorithme de Sliced Wasserstein

B.1 Sans régularisation



FIGURE 5 – Tranfert de couleur par Sliced Wasserstein Distance

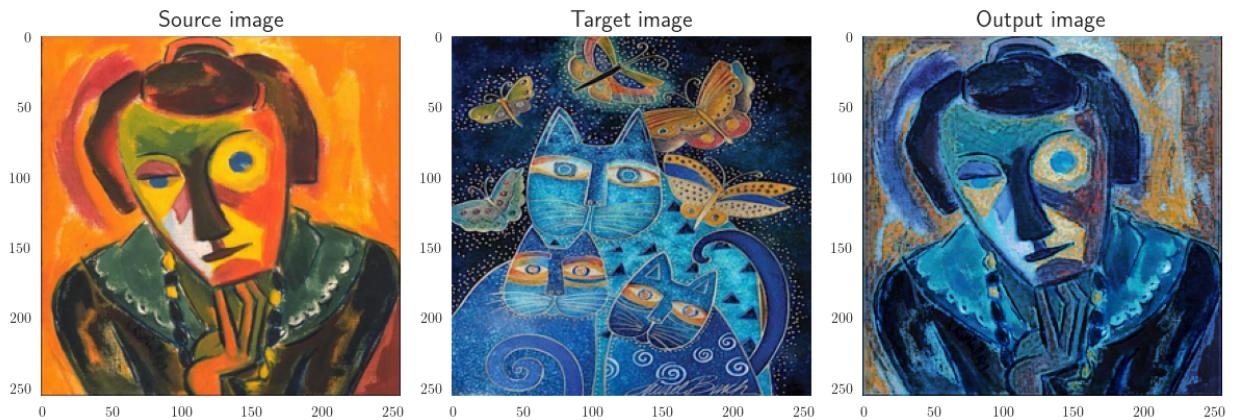


FIGURE 6 – Un autre exemple de tranfert de couleur par Sliced Wasserstein Distance

B.2 Sortie de l'algorithme après régularisation

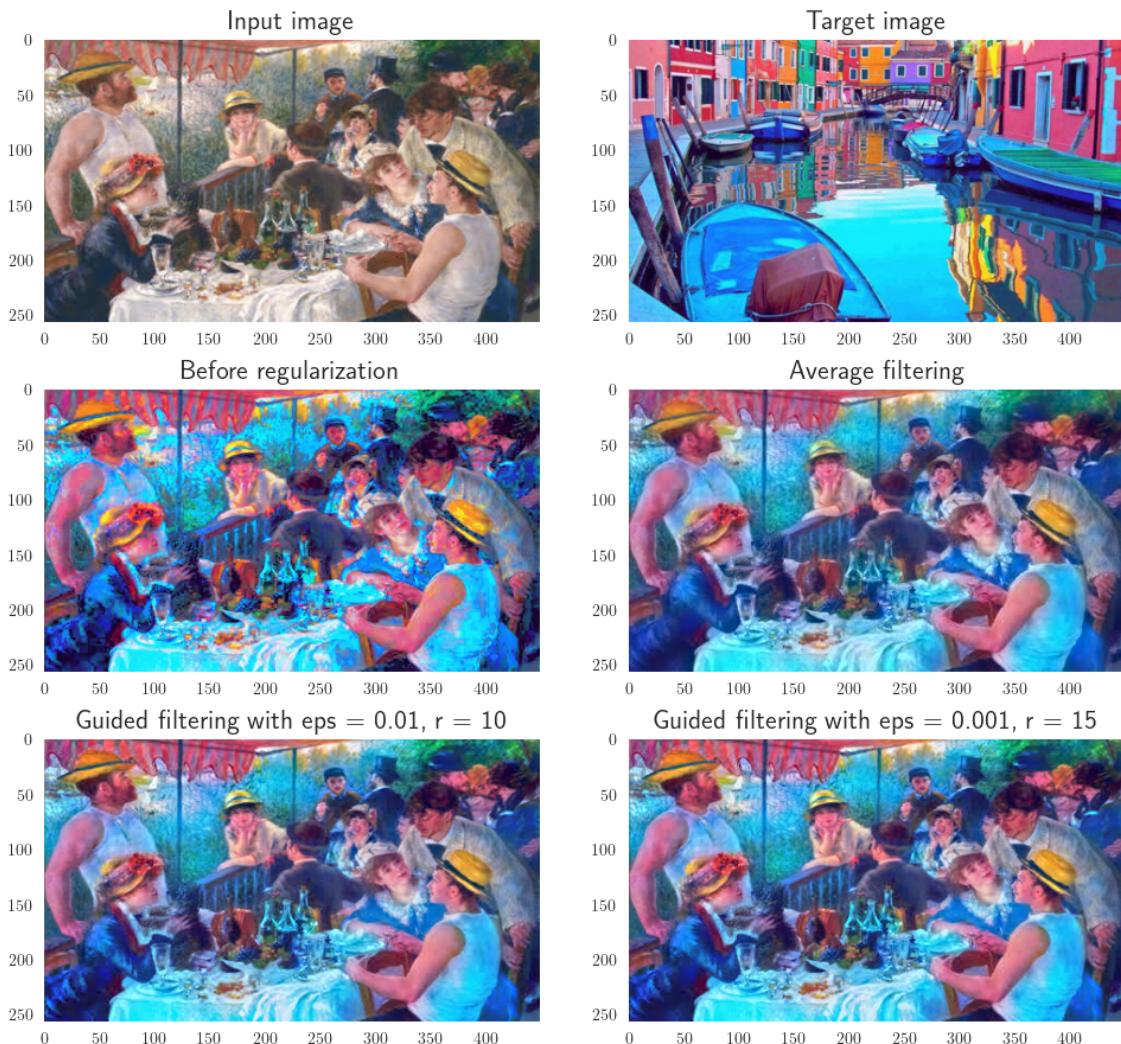


FIGURE 7 – Résultat par Sliced Wasserstein Distance et lissage par différents filtres

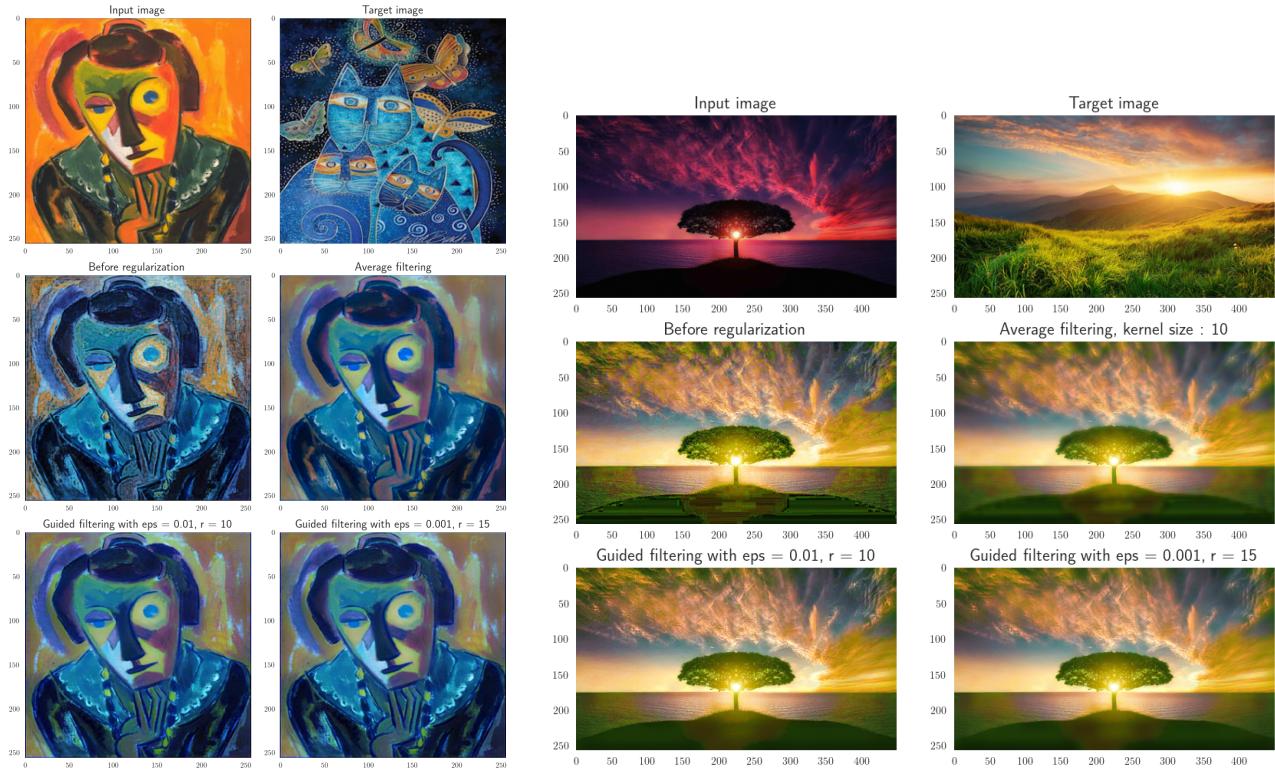


FIGURE 8 – Résultat par Sliced Wasserstein Distance et lissage par différents filtres sur 2 autres exemples. (Zoom in for better resolution)

Références

- [1] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(6) :1397–1409, 2013.
- [2] Nicolas Papadakis Julien Rabin, Sira Ferradans. Adaptive color transfer with relaxed optimal transport. 2014.
- [3] Gabriel Peyré and Marco Cuturi. Computational optimal transport. 2018.