

## Edition d'image de Poisson

Tanguy RENAUDIE, Mehmet BASAGAC, Alexandre GAVAUDAN,  
Timothe SCHMIDT

26 juillet 2022

1 Présentation du projet

2 Modélisation mathématique

3 Modélisation numérique

4 Disparition de fragments

5 Mélange d'images

- Gradient mixte
- Gradient max

## 1 Présentation du projet

## 2 Modélisation mathématique

## 3 Modélisation numérique

## 4 Disparition de fragments

## 5 Mélange d'images

- Gradient mixte
- Gradient max

## Contexte

Restaurer des portions inconnues ou endommagées de façon crédible est un problème important en traitement d'image.

## Exemple

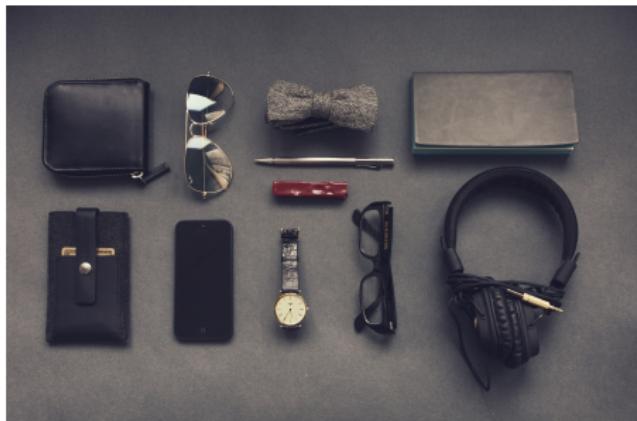


Figure – Image à modifier

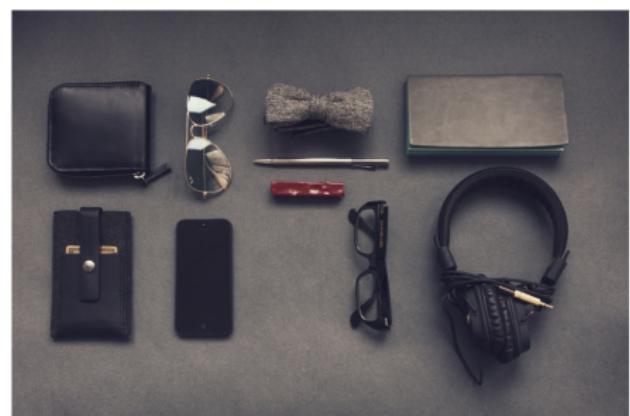


Figure – Image sans les lunettes

# Démo

Démonstration du programme

## Exemple d'application de notre algorithme



Figure – Arc-en-ciel



Figure – Paysage montagneux

## Exemple d'application de notre algorithme



Figure – Image sans les lunettes

# Objectifs

- ➊ Modéliser mathématiquement le problème
- ➋ Proposer des algorithmes de résolution numérique
- ➌ Comparer les performances des algorithmes

1 Présentation du projet

2 Modélisation mathématique

3 Modélisation numérique

4 Disparition de fragments

5 Mélange d'images

- Gradient mixte
- Gradient max

# Introduction

## Hypothèses

- ➊ L'image cible est une fonction  $\phi$  à valeurs dans  $[0,1]$  définie sur un domaine  $\Omega$
- ➋ On cherche une fonction  $\psi$  qui correspond exactement à  $\phi$  sur  $\partial\Omega$  et dont la texture correspond approximativement à celle d'une troisième image source  $\chi$  sur  $\Omega$
- ➌ La texture de l'image est principalement contenue dans le **gradient**

## Position du problème

Ainsi on cherche à minimiser :

$$\min_{\psi} \int_{\Omega} \|\nabla(\psi - \chi)\|^2 \text{ avec } \psi|_{\partial\Omega} = \phi|_{\partial\Omega}$$

En posant  $u = \psi - \chi$  et  $f = (\phi - \chi)|_{\partial\Omega}$ , on se retrouve avec une fonction  $u$  qui est solution du **problème variationnel de Dirichlet** : la minimisation de l'**énergie de Dirichlet** de  $u$ , avec les conditions aux bords imposées par  $f$  :

$$\min_u \int_{\Omega} \|\nabla u\|^2 \text{ avec } u|_{\partial\Omega} = f.$$

Cela revient aussi à chercher  $u$  comme solution du **problème de Dirichlet** :

$$\Delta u = 0 \text{ sur } \Omega \text{ with } u|_{\partial\Omega} = f$$

# Démonstration

On considère une variation  $\epsilon$  autour du minimiseur  $u$ , on a alors

$$\int_{\Omega} \|\nabla(u + \epsilon)\|^2 = \int_{\Omega} \|\nabla u\|^2 + 2 \int_{\Omega} \nabla u \cdot \nabla \epsilon + \int_{\Omega} \|\epsilon\|^2$$

Ainsi,  $u$  est un minimiseur si et seulement si

$$\forall \epsilon \text{ tel que } \epsilon|_{\partial\Omega} = 0, \int_{\Omega} \nabla u \cdot \nabla \epsilon = 0$$

Ainsi, d'après la première identité de Green, cette condition ne tient que si  $\Delta u = 0$  sur  $\Omega$ .



## Cadre général

On considère plus généralement :

$$\min_{\psi} \int_{\Omega} \|\nabla \psi - \mathbf{v}\|^2 \text{ avec } \psi|_{\partial\Omega} = \phi|_{\partial\Omega}$$

où  $\mathbf{v}$  représente un champ de gradient vectoriel différentiable sur  $\Omega$ .  
Alors,  $\psi$  doit maintenant vérifier l'équation d'Euler-Lagrange associée

$$\Delta \psi = \operatorname{div}(\mathbf{v}) \text{ sur } \Omega \text{ avec } \psi|_{\partial\Omega} = \phi|_{\partial\Omega}$$

## Commentaires de suivi

Nous avons pris du temps à comprendre l'équation mathématique.

**Difficulté 1 :** La notation  $\partial\Omega$  ne correspond pas à la frontière au sens topologique, mais plutôt à l'ensemble des points de  $\Omega$  dont l'un des quatre voisins est dans  $\Omega^c$ . Ces points sont directement donnés par la contrainte du bord.

**Difficulté 2 :** Certains points ne sont pas dans  $\partial\Omega$ , mais possèdent un voisin dans  $\partial\Omega$ . Ces points  $x$  ont une équation du type :

$$\sum_{y \in Vois(x) \cap \partial\Omega} \Psi(y) + \sum_{y \in Vois(x) \cap \Omega \setminus \partial\Omega} \Psi(y) - 4\psi(x) = \sum_{y \in Vois(x)} \chi(y) - 4\chi(x)$$

Il nous a donc fallu comprendre la différence entre ces trois types de points au sein de  $\Omega$ .

1 Présentation du projet

2 Modélisation mathématique

3 Modélisation numérique

4 Disparition de fragments

5 Mélange d'images

- Gradient mixte
- Gradient max

# Solution Numérique

Déterminer la fonction  $\psi$  sur  $\Omega$  telle que :

$$\begin{aligned}\Delta\psi(x) &= \operatorname{div}(\mathbf{v}), \quad \forall x \in \Omega \\ \psi|_{\partial\Omega} &= \phi|_{\partial\Omega}\end{aligned}$$

Première approche : avec  $\mathbf{v} = \nabla\chi$

$$\begin{aligned}\Delta\psi(x) &= \Delta\chi(x), \quad \forall x \in \Omega \\ \psi|_{\partial\Omega} &= \phi|_{\partial\Omega}\end{aligned}$$

## Discrétisation du domaine $\Omega$

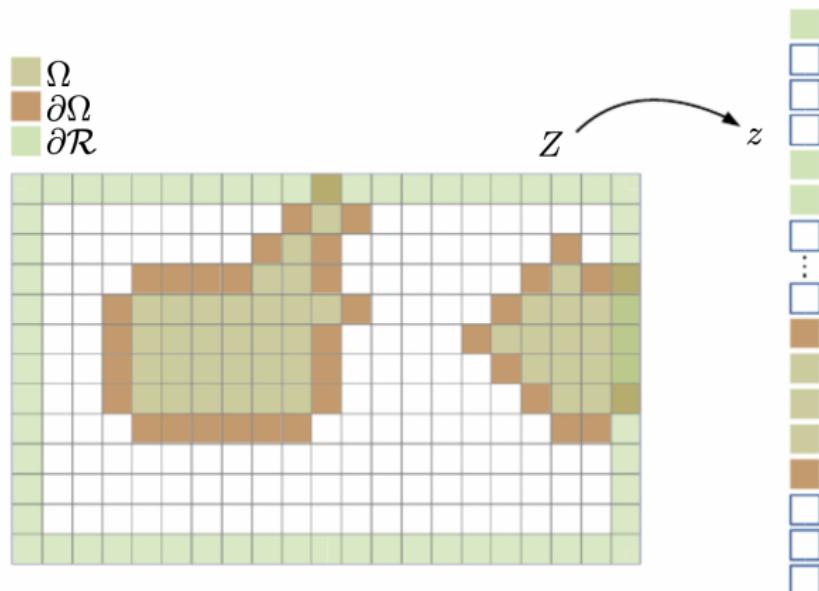


Figure – Discrétisation du laplacien, *Poisson Image Editing*, Martino, J. et al.,  
2016/11

# Méthode des différences finies

Pour une fonction deux fois différentiable  $u$ ,

$$\Delta\psi(x_i, x_j) = \frac{\partial^2\psi}{\partial x^2}(x_i, x_j) + \frac{\partial^2\psi}{\partial y^2}(x_i, x_j) \quad (1)$$

$$\psi(x_{i+1}) = \psi(x_i) + \psi'(x_i)h + \psi''(x_i)\frac{h^2}{2} + O(h^3)$$

$$\psi(x_{i-1}) = \psi(x_i) - \psi'(x_i)h + \psi''(x_i)\frac{h^2}{2} + O(h^3)$$

$$\psi''(x_i) = \frac{\psi(x_{i+1}) + \psi(x_{i-1}) - 2\psi(x_i)}{h^2} + O(h^2)$$

Laplacien en 2D :

$$\Delta\psi(x_{i,j}) = \frac{\psi(x_{i+1,j}) + \psi(x_{i-1,j}) + \psi(x_{i,j+1}) + \psi(x_{i,j-1}) - 4\psi(x_{i,j})}{h^2} + O(h^2)$$

# Méthode des différences finies

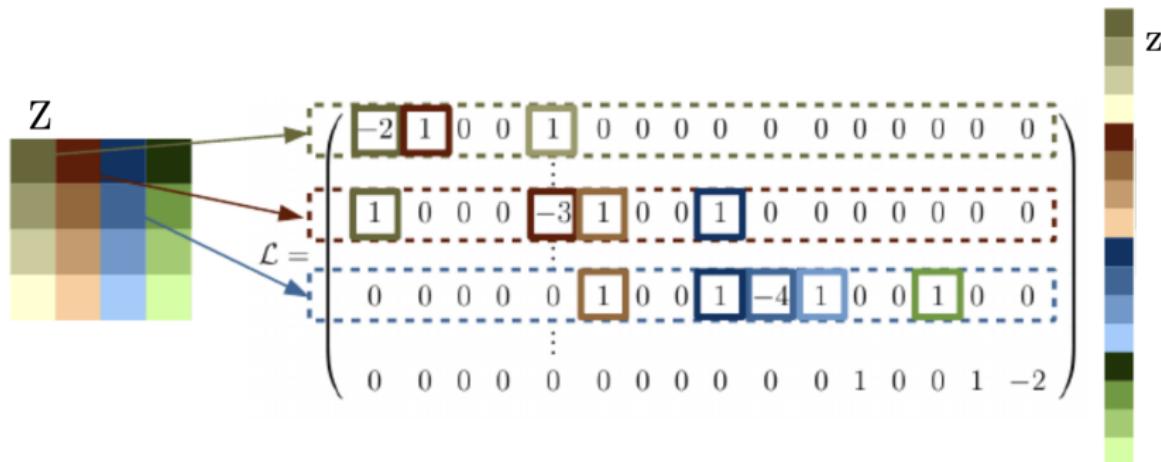


Figure – Discréétisation de  $\Omega$ , *Poisson Image Editing*, Martino, J. et al., 2016/11

# Méthode des différences finies

On introduit la matrice opérateur Laplacien discret :

$$A = \begin{pmatrix} -4 & 1 & & & 1 \\ 1 & -4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -4 & 1 \\ 1 & 1 & & -4 & 1 \\ & 1 & & 1 & -4 & 1 \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -4 & 1 \\ & & & & 1 & -4 \end{pmatrix} \in \mathbb{R}^{N \times N}$$

# Première approche

$$\Delta\psi = \Delta\chi \text{ avec } \psi|_{\partial\Omega} = \phi|_{\partial\Omega}$$

$$\begin{aligned}
 & \left( \begin{array}{c} \psi_{1,1} \\ \psi_{1,2} \\ \vdots \\ \psi_{1,m} \\ \psi_{2,1} \\ \psi_{2,2} \\ \vdots \\ \psi_{2,m} \\ \vdots \\ \psi_{n,1} \\ \vdots \\ \psi_{n,m} \end{array} \right) \in \mathbf{R}^N \text{ et } t = \left( \begin{array}{c} \chi_{1,1} \\ \chi_{1,2} \\ \vdots \\ \chi_{1,m} \\ \chi_{2,1} \\ \chi_{2,2} \\ \vdots \\ \chi_{2,m} \\ \vdots \\ \chi_{n,1} \\ \vdots \\ \chi_{n,m} \end{array} \right) \in \mathbf{R}^N \\
 & \text{devient : } A\mathbf{x} = A\mathbf{t} \text{ où } u =
 \end{aligned}$$

## Problème : bords $\partial\Omega$

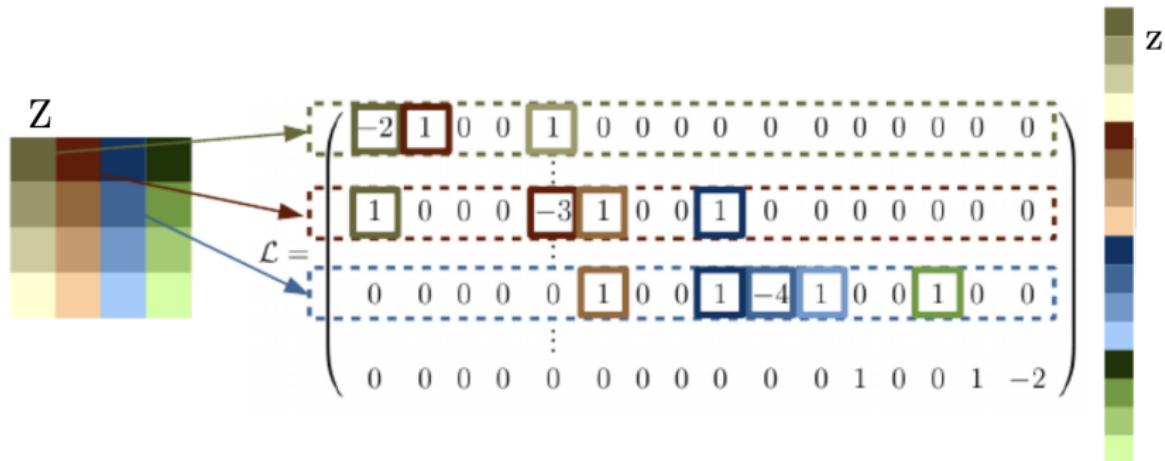


Figure – Discrétisation du laplacien, *Poisson Image Editing*, Martino, J. et al., 2016/11

## Premiers résultats

Divergences de pixels et temps de calculs trop longs



Figure – Panda collé dans une forêt

## Commentaires de suivi

Pour ajouter la contrainte aux bords, notre première approche consistait à poser

$$\forall i \in [1, n], A_{i*m+i} = 1$$

$$\forall i \in [1, n], j \neq i, A_{i*m+j} = 0$$

Ceci impose bien la contrainte au bord  $\psi|_{\partial\Omega} = \phi|_{\partial\Omega}$

Cette première modélisation était à la fois trop longue à calculer, et donnait lieu à des divergences de couleurs de pixels. Pour le problème de temps, nous avons eu recours à des matrices creuses, en utilisant la librairie Scipy Sparse. Ceci permettait de réduire les temps de calcul par deux environ, car les opérateurs laplacien sont des matrices pleines de zéros.

Nous avons également changé les équations, pour réduire les temps de calcul et diminuer les divergences. Ce premier système linéaire prenait pour variable à la fois les points du bord et les points pas dans le bord. Or, les points du bord ne sont plus des variables, car ils sont donnés par l'équation du bord. Donc on peut réduire le système linéaire à  $\Omega \setminus \partial\Omega$ . Ce nouveau système d'équation fonctionne mieux.

# Implémentation des matrices creuses

0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	8	5	
4	0	0	5	0	0	0	7	
0	0	0	9	5	0	0	2	
0	0	0	1	0	4	0	0	
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
0	8	0	0	0	0	0	0	0

(a)

val	4	2	3	1	8	5	9	1	5	4	8	5	7	2
row_idx	3	6	7	1	8	3	4	5	4	5	2	2	3	4
col_ptr	1	4	6	6	9	10	11	12	15					

(b)

Figure – Dense vs Sparse Matrices

## Modification des équations

$$\Delta\psi = \Delta\chi, \text{ avec } \psi|_{\partial\Omega} = \phi|_{\partial\Omega}$$

est en fait équivalent à :

$$\Delta(P_\Omega\psi + P_{\partial\Omega}\psi) = \Delta\chi, \text{ avec } \psi|_{\partial\Omega} = \phi|_{\partial\Omega}$$

L'équation devient :

$$\Delta P_\Omega\psi = \Delta\chi - \Delta P_{\partial\Omega}\phi$$

# Opérateurs codés

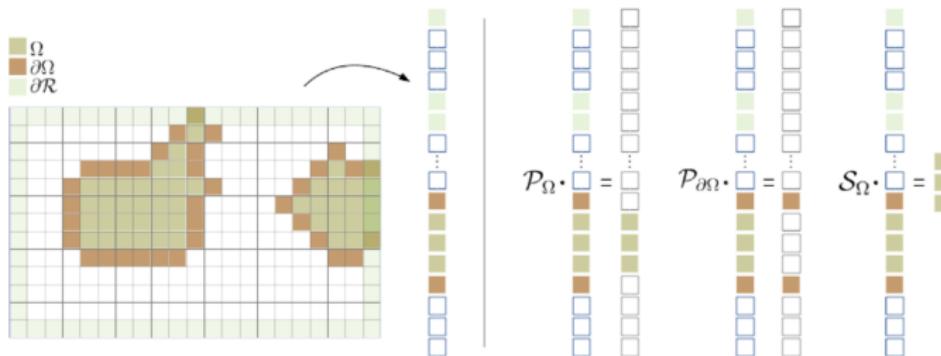


Figure – Opérateurs de projection

$$SAS^T \mathbf{u} = SAs - SAR$$

## Deuxièmes résultats

Moins de divergences, et un temps de calcul moins long



Figure – Panda collé dans une forêt

## Commentaires de suivi

Pour mieux contrôler les valeurs du champ de gradient  $\mathbf{v}$ , nous avons dû implémenter des opérateur de gradient, par rapport à  $x$  et par rapport à  $y$ , notés  $D_x$  et  $D_y$ . Ceci nous donnait la possibilité de choisir la texture (le gradient) de l'image finale, dans le domaine  $\Omega \setminus \partial\Omega$ .

**Méthode 1 : Gradient Mix** - mélanger les gradients de l'image source et de l'image de destination. Nous pouvons choisir une combinaison linéaire  $\mathbf{v} = a\nabla + (1 - a)\nabla\Phi$ . Nous avons vu que cela ne fonctionnait pas très bien pour superposer les deux images, car cela conserve tous les éléments de chaque image.

**Méthode 2 : Max Gradient** - choisir comme valeur de  $\mathbf{v}$ , pour chaque pixel, le maximum du gradient de l'image source et de l'image de destination :

$$\mathbf{v} = \max\{\nabla\chi, \nabla\Phi\}$$

Ceci fonctionne bien mieux. Et donne les résultats ci-dessous. On obtient bien la fonctionnalité "Glue" de Photoshop. Choisir le minimum entre les deux gradients était aussi une possibilité, et permettait de gommer les défauts d'une image. C'est la fonction "Erase" de Photoshop.

# Opérateurs de gradient codés

$$D_x = \begin{pmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{pmatrix}$$

Figure – Opérateurs dérivation par rapport à x

$$SAS^T \mathbf{u} = S(D_x \mathbf{v_x} + D_y \mathbf{v_y} - A\mathbf{r})$$

1 Présentation du projet

2 Modélisation mathématique

3 Modélisation numérique

4 Disparition de fragments

5 Mélange d'images

- Gradient mixte
- Gradient max

Gradient source :  $\mathbf{v} = \nabla \chi$

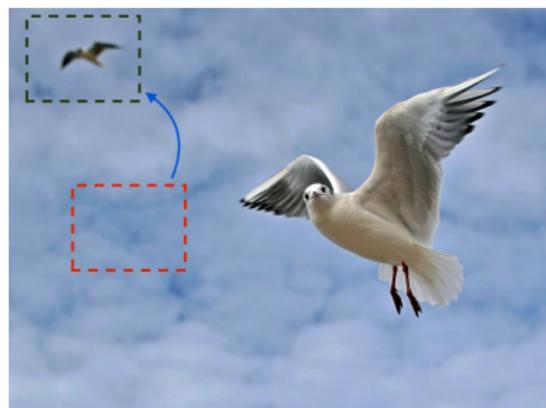


Figure – Photo à corriger



Figure – Photos retouchées après une seule correction puis après 3 corrections

Gradient min :  $\mathbf{v} = \text{Min} \{\nabla \chi, \nabla \phi\}$



Figure – Photo retouchée après une seule correction gradient min

## 1 Présentation du projet

## 2 Modélisation mathématique

## 3 Modélisation numérique

## 4 Disparition de fragments

## 5 Mélange d'images

- Gradient mixte
- Gradient max

Gradient mixte :  $\mathbf{v} = \alpha \nabla \chi + \beta \nabla \phi$  avec  $\alpha + \beta = 1$



Figure – Forêt

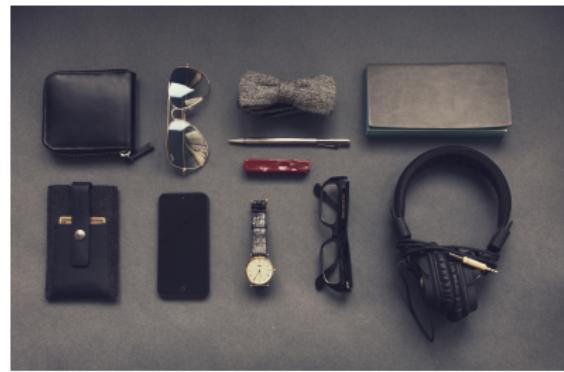


Figure – Objects divers

Gradient mixte :  $\mathbf{v} = \alpha \nabla \chi + \beta \nabla \phi$  avec  $\alpha + \beta = 1$



Figure – Gradient mixte ( $\alpha = \beta = 0.5$ )

# limite du gradient mixte

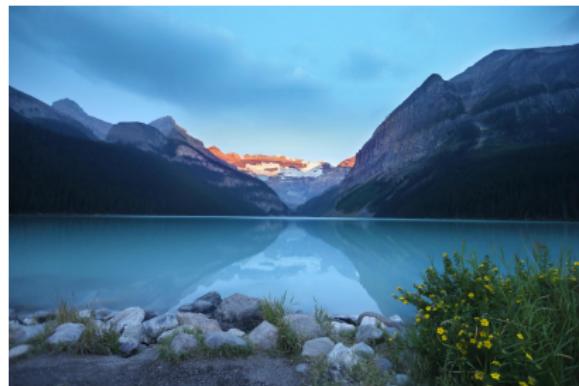


Figure – Paysage montagneux



Figure – Soleil

# Limite du gradient mixte

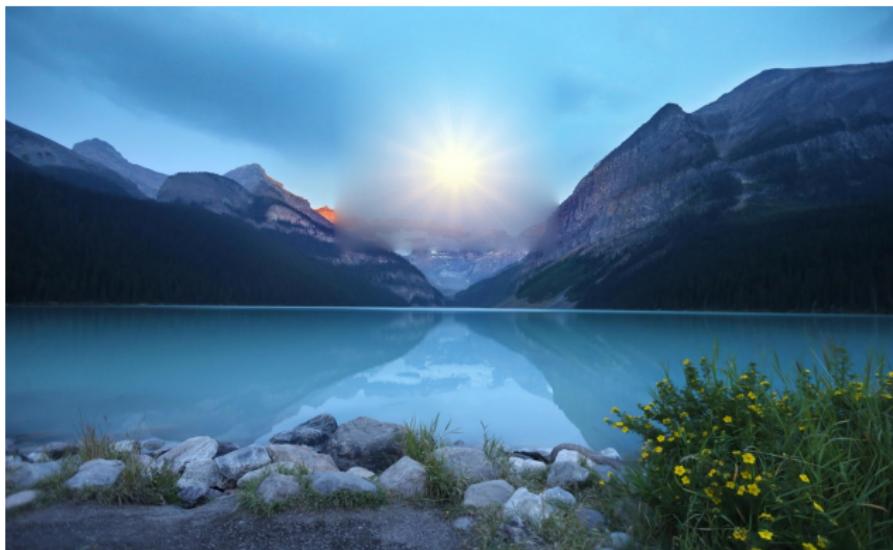


Figure – Gradient mixte inadapté ( $\alpha = 0.7$  et  $\beta = 0.3$ )

Gradient max :  $\mathbf{v} = \text{Max} \{ \nabla \chi, \nabla \phi \}$



Figure – Lunettes cachées dans une forêt

Gradient max :  $\mathbf{v} = \text{Max} \{ \nabla \chi, \nabla \phi \}$



Figure – Gradient mixte ( $\alpha = \beta = 0.5$ )



Figure – Gradient max

Gradient max :  $\mathbf{v} = \text{Max} \{ \nabla \chi, \nabla \phi \}$

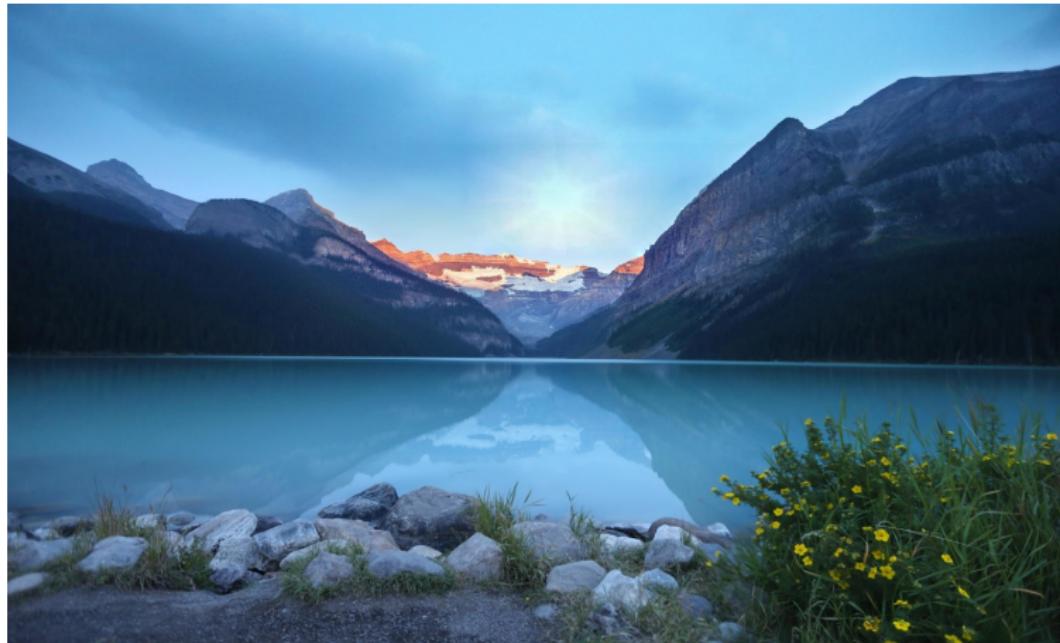


Figure – Soleil rasant

Gradient max :  $\mathbf{v} = \text{Max} \{ \nabla \chi, \nabla \phi \}$



Figure – Gradient mixte ( $\alpha = \beta = 0.5$ )



Figure – Gradient max

Gradient max :  $\mathbf{v} = \text{Max} \{\nabla\chi, \nabla\phi\}$



Figure – Requin en eaux profondes



Figure – Eaux peu profondes

Gradient max :  $\mathbf{v} = \text{Max} \{\nabla\chi, \nabla\phi\}$



Figure – Requin dans des eaux peu profondes

Gradient max :  $\mathbf{v} = \text{Max} \{\nabla \chi, \nabla \phi\}$



Figure – Nuée d'oiseaux



Figure – Parachutiste

Gradient max :  $\mathbf{v} = \text{Max} \{ \nabla \chi, \nabla \phi \}$



Figure – Parachutiste accompagné d'oiseaux

Gradient max :  $\mathbf{v} = \text{Max} \{\nabla\chi, \nabla\phi\}$



shutterstock.com · 1208669053

Figure – Bateau qui sombre



Figure – Piscine

Gradient max :  $\mathbf{v} = \text{Max} \{ \nabla \chi, \nabla \phi \}$



Figure – Bateau qui sombre dans une piscine

## Limite gradient max



Figure – gradient max inadapté

## Améliorations possibles

Nous pouvons améliorer ce projet de traitement d'image, avec les méthodes suivantes :

- **la librairie Cython**, pour augmenter la vitesse de l'algorithme.
- **le broadcasting numpy** sur les trois valeurs RGB, pour réduire la complexité en calculs, plutôt qu'une séparation de l'algorithme en trois.
- **utilisation d'une méthode de Fourier**, qui consiste à sélectionner les fréquences qui ressortent le plus entre la source et la destination.

# Bibliographie

Martino, J. Facciolo, Gabriele Meinhardt-Holzapfel, Enric. (2016). Poisson Image Editing. *Image Processing On Line*. 5. 300-325.  
[10.5201/ipol.2016.163](https://doi.org/10.5201/ipol.2016.163).