

Rendu TP 2 : Équation de Poisson

EL OUAIFI, Moussa

2 mars 2021

1 Retinex Poisson Equation

Avant de commencer, lisez attentivement cet article (la description de la méthode fait moins de 2 pages et elle est très simple).

1.1. Quel est le problème que la méthode cherche à résoudre ?

L'objectif de la méthode Retinex est de mimiquer la perception visuelle de l'homme, donc la méthode améliore les effets de la perception visuelle de l'homme sans améliorer l'image. Le modèle PDE-Retinex consiste à aplatiser les couleurs dans les régions ayant un gradient de couleurs faible. Ceci permet d'éliminer les petites variations d'intensité à cause de l'existence des nuances, donc le paramètre t de l'algorithme est pris en général $t < 10$ pour ne pas enlever des détails importants. Cette méthode a ses limites, en effet elle n'est pas si efficace dans le cas des images très sombres et de mauvaise qualité.

1.2. Commenter chaque pas de l'algorithme.

Pour chaque chaîne de couleur (RGB) :

- **pas 0 :** input le seuil t , l'image originale.

L'algorithme prend comme paramètre "t" et image "l'image originale".

- **pas 1 : Résoudre l'équation différentielle discrète avec la condition au bord de Neumann, $-\Delta_d u(i, j) = F(i, j)$:**

1. : Etendre l'image originale, $I = \{0, \dots, M-1\} \times \{0, \dots, N-1\}$, de dimensions (N, M) à une image de dimensions $(2N, 2M)$ par effet miroir de la manière suivante :

Symétrisation à droite :

$$\forall j \in \{0, \dots, N-1\}, \forall i \in \{1, \dots, M\}, I(M+i-1, j) = I(M-i, j).$$

Symétrisation en bas :

$$\forall i \in \{0, \dots, 2M-1\}, \forall j \in \{1, \dots, N\}, I(i, -j) = I(i, j-1).$$

Remarque : L'intérêt de cette étape est d'avoir une transformée de Fourier Discrète réelle, pour ce faire on crée une image symétrique. En effet, Rappelons *qu'un échantillon réel a sa transformée de Fourier discrète réelle si et seulement si l'échantillon est symétrique*.

2. On calcule les valeurs $F(k, l)$:

$F(k, l) = f(I(k, l) - I(k + 1, l)) + f(I(k, l) - I(k - 1, l)) + f(I(k, l) - I(k, l - 1)) + f(I(k, l) - I(k, l + 1))$, où la fonction f est définie comme suit : $f(x) = 0$, si $|x| \leq t$, f égale l'identité sinon. Avec $I = \{0, \dots, 2M - 1\} \times \{-N, \dots, N - 1\}$ est la "nouvelle" l'image à traiter.

3. On applique la Transformation de Fourier Discrète à l'équation de poisson "discrète" : $-\Delta_d u(i, j) = F(i, j)$ sur chaque chaîne (u est rouge, Vert et Blue en chaque étape).

L'équation devient équivalente à :

$$\hat{u}(k, l)(4 - 2\cos(\frac{2\pi k}{2N}) - 2\cos(\frac{2\pi l}{2M})) = \hat{F}(k, l)$$

Donc pour tous $(k, l) \neq (0, 0)$:

$$\hat{u}(k, l) = \frac{\hat{F}(k, l)}{4 - 2\cos(\frac{2\pi k}{2N}) - 2\cos(\frac{2\pi l}{2M})}$$

Remarque : Ici, F est bien réel et symétrique, donc \hat{F} est réelle et donc \hat{u} aussi. (même en $\hat{u}(0, 0) = \frac{1}{4NM} \sum_0^{2N-1} \sum_0^{2M-1} u(k, l)$, la moyenne de u qui est réelle.). Donc créer une image symétrique par effet miroir nous a permis d'obtenir un échantillon réelle, comme nous l'avons mentionné dans une remarque précédente.

4. On trouve les valeurs de u par la transformation de Fourier Inverse :

$$u(n, m) = \sum_0^{2N-1} \sum_0^{2M-1} \hat{u}(k, l) e^{i \frac{2\pi kn}{2N}} e^{i \frac{2\pi lm}{2M}}$$

5. On normalise les valeurs de u de manière qu'elles ont la moyenne et la variance de l'image d'origine.
6. On limite les valeurs de u à l'intervalle $[0, 255]$:

Les valeurs de u qui sortent de cet intervalle sont saturées à 0 (si $u < 0$) et 255 (si $u > 255$).

— **pas 2 : Output** Retourner la solution image.

1.3. Quel est l'effet de cet algorithme quand $t = 0$?

Si $t = 0$ l'algorithme retourne l'image initiale ; En effet, $t = 0$ implique que la fonction f définie en paragraphe 1.2. est égale à l'identité ; donc les valeurs $F(k, l)$ sont données par :

$$F(k, l) = 4I(k, l) - I(k, l + 1) - I(k, l - 1) - I(k + 1, l) - I(k - 1, l).$$

Donc la variation d'intensité reste la même et identique à la perception visuelle de l'homme d'image I . L'algorithme n'a pas un véritable effet.

1.4. Quel est l'effet de cet algorithme quand $t = \infty$?

Si $t = \infty$, la fonction f est identiquement nulle par suite les fonctions F, \hat{F} sont nulles, on aura donc \hat{u} identiquement nulle, ce qui donne une fonction u nulle sur toutes les chaînes (Rouge, vert et bleu).

L'image perdra donc tous ses détails, l'algorithme retournera donc une image noire.

1.5. Appliquer l'algorithme sur une image de votre choix avec des valeurs différentes de t (très grands et très petits), et commentez le résultat. Veuillez choisir une image intéressante qui sert à illustrer l'effet du paramètre t .

La figure 1 illustre bien l'effet de varier le paramètre t .

Dans l'image originale, les carreaux A et B ont même niveau du gris 120. Mais dans l'image traitée par $t = 3$, le careau B (niveau du gris 160) est plus brillant que A (niveau du gris 120) le contraste de le carreau B a augmenté d'une manière respectant la perception visuelle de l'homme. Le même commentaire est valide pour le cas de $t = 5$ et $t = 10$.

La deuxième remarque concerne le coté droit du cylindre, plus que la valeur de t augmente, plus que ce coté perd plus de détails et devient plus sombre, ce qui est prévisible car cette zone est caractérisée par une variation d'intensité importante, qui dépasse la valeur de paramètre t .

Finalement, on voit clairement que choisir un t très grand qui tend vers l'infini, élimine beaucoup de détails de l'image initiale et donne une image plus sombre.

1.6. Considérer la formule qui définit F à partir de I (section 2 de l'article) pour le cas particulier $t = 0$. Cette formule peut s'interpréter comme l'approximation par différences finies d'un opérateur différentiel linéaire agissant sur I . Lequel ?

Quitte à considérer un développement limité, on travaille par l'approximation du milieu continu (on suppose que la distance inter pixels est très minime, de sorte que la fonction u varie peu sur

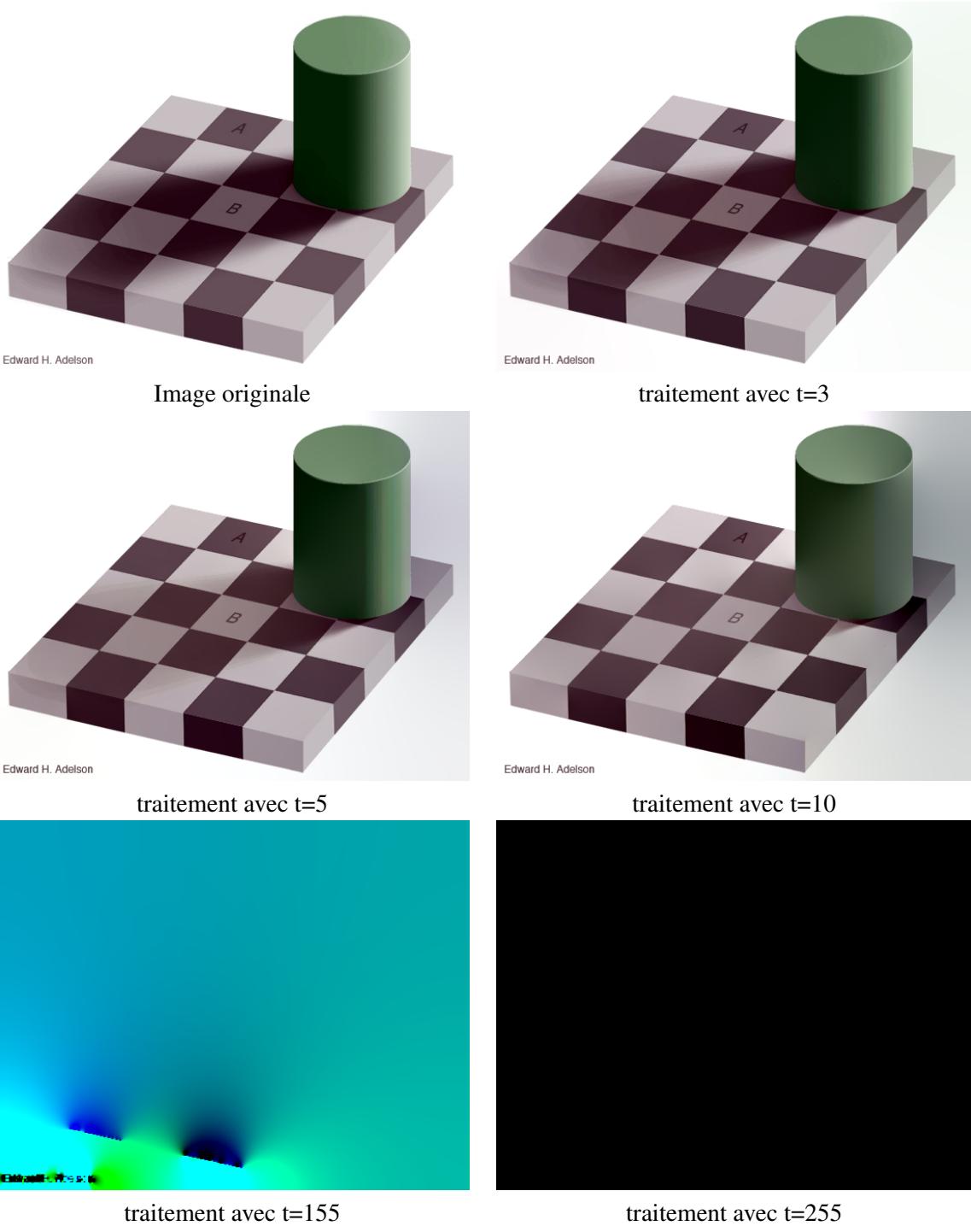


FIGURE 1 – Résultat d’algorithme Retinex avec différentes valeurs t .

deux pixels voisins) :

Pour $t = 0$ on a :

$$F(k, l) = [I(k, l) - I(k, l+1)] + [I(k, l) - I(k, l-1)] + [I(k, l) - I(k+1, l)] + [I(k, l) - I(k-1, l)]$$

$$F(k, l) = [-\frac{\partial I}{\partial l}(k, l+1)] + [\frac{\partial I}{\partial l}(k, l)] + [-\frac{\partial I}{\partial k}(k+1, l)] + [\frac{\partial I}{\partial k}(k, l)]$$

$$F(k, l) = -\frac{\partial^2 I}{\partial l^2}(k, l) - \frac{\partial^2 I}{\partial k^2}(k, l), \text{ Donc :}$$

$$F(k, l) = -\Delta I(k, l)$$

L'opérateur est donc $-\Delta$, avec Δ est le **Laplacien**.

1.7. Écrire une version continue de la méthode proposée par l'article. Soit $I : [0, 2\pi]^2 \rightarrow \mathbf{R}$ l'image d'entrée et f la fonction de seuillage. Écrire la formule qui donne l'image résultat u à partir de I et f .

La méthode sera la même, sauf qu'on va pas passer du discret au continu. Donc on remplace le Laplacien discrète Δ_d par le laplacien usuel Δ et la Transformation de Fourier discrète et son inverse par la transformation de Fourier continue.

- On commence par rendre l'image symétrique par effet miroir pour obtenir une image $I : [0, 4\pi] \times [-2\pi, 2\pi] \rightarrow \mathbf{R}$.
- $F(x, y) = f(I(x, y) - I(x^-, y)) + f(I(x, y) - I(x^+, y)) + f(I(x, y) - I(x, y^-)) + f(I(x, y) - I(x, y^+))$ avec $f(x) = 0$, si $|x| \leq t$, f égale à l'identité sinon.

- L'équation de poisson correspondante sera :

$$-\Delta u(x, y) = F(x, y)$$

- On calcule \hat{F} , la transformation "continue" de Fourier de F :

$$\hat{F}(a, b) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} F(x, y) e^{-i2\pi(xa+yb)} dx dy$$

- la transformation de Fourier appliquée à l'équation de Poisson :
 $(x^2 + y^2)\hat{u}(x, y) = \hat{F}(x, y)$, Donc pour tous $(x, y) \neq (0, 0)$:

$$\hat{u}(x, y) = \frac{\hat{F}(x, y)}{x^2 + y^2}$$

- On obtient u par transformation de Fourier inverse :

$$u(a, b) = \frac{1}{(2\pi)^2} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \hat{F}(x, y) e^{i2\pi(xa+yb)} dx dy$$

- On normalise u pour que l'image obtient la même moyenne et variance, et on restreint les résultats au domaine d'image $[0, 2\pi] \times [0, 2\pi]$ en saturant les valeurs de u hors cet intervalle.

Remarque : on travaille sous l'hypothèse que la fonction u est continue.

1.8. Comparer la formule (1) de l'article et la formule (5.22) du polycopié. Ces deux formules visent à résoudre le même problème. Lequel ? Pourquoi sont-elles si différentes ?

Les deux formules cherchent à trouver les coefficients de Fourier de u (et vérifier qu'ils sont réels) à partir des coefficients de Fourier de F , à l'exception de $u(0,)$ qui est la moyenne de u .

Les deux formules sont différentes parce qu'on travaille techniquement sur deux fonctions différentes. En effet la fonction u dans la formule (5.22) est une fonction trigonométrique donc lisse et du coup lorsqu'on applique la transformation de Fourier on trouve la formule usuelle de la transformation de Fourier de dérivée d'une fonction. Dans la formule (1), la transformation de Fourier n'est pas appliquée sur le laplacien de la fonction u mais sur une approximation linéaire de ce dernier, ce qui nous donne des coefficients un peu différents de celle de la formule (5.22).

1.9. Est-il possible d'interpréter la formule (1) comme une approximation numérique de la formule (5.22) ?

On utilise la formule de Taylor avec reste intégral on obtient :

$$\cos\left(\frac{2\pi k}{N}\right) = 1 - \frac{1}{2}\left(\frac{2\pi k}{N}\right)^2 + \frac{1}{2} \int_0^{\frac{2\pi k}{N}} \left(\frac{2\pi k}{N} - t\right)^3 \sin(t) dt$$

$$\cos\left(\frac{2\pi l}{M}\right) = 1 - \frac{1}{2}\left(\frac{2\pi l}{M}\right)^2 + \frac{1}{2} \int_0^{\frac{2\pi l}{M}} \left(\frac{2\pi l}{M} - t\right)^3 \sin(t) dt$$

Par suite,

$$4 - 2\cos\left(\frac{2\pi k}{N}\right) - 2\cos\left(\frac{2\pi l}{M}\right) = \left(\frac{2\pi k}{N}\right)^2 + \left(\frac{2\pi l}{M}\right)^2 - \int_0^{\frac{2\pi k}{N}} \left(\frac{2\pi k}{N} - t\right)^3 \sin(t) dt - \int_0^{\frac{2\pi l}{M}} \left(\frac{2\pi l}{M} - t\right)^3 \sin(t) dt$$

Donc pour des valeurs $l < N$ et $k < l$ choisies de sorte que le reste intégral est strictement positive et non négligeable, **on ne peut pas** interpréter la formule (1) comme une approximation numérique de la formule (5.22).

***1.10.** Imaginer une variante de cet article, mais en utilisant la formule (5.22) du polycopié au lieu de la formule (1) de l'article. Croyez-vous que les résultats seraient visuellement similaires, sur une image typique ? Décrire ou exhiber une image I pour laquelle les résultats des deux variantes seraient aussi différents que possible.

Les résultats ne seraient pas visuellement similaires sur toute image, surtout si on choisit un paramètre t très petit. En effet si on travaille sur une image dont les bords sont saturés (par exemple une image qui est le résultat de l'algorithme Simplest Color Balance SCB), on va obtenir une variation d'intensité différente pour chacune des deux méthodes. L'idée derrière cette différence dans la variation de l'intensité est que puisqu'on n'a pas une approximation numérique "acceptable", on aura certainement une différence entre la variation de l'intensité de certaines mêmes pixels des deux images, cette différence conduit à des résultats largement différents si on choisit t très petit.

2 Poisson Image Editing

Pour cet article, on considère seulement la partie *seamless cloning* de la demo en ligne. Cette partie correspond au chapitre sur l'éditeur de Poisson du polycopié.

La demo propose deux modélisations : l'équation de Poisson sur le domaine entier de l'image (qui est résolue ensuite par technique d'analyse de Fourier), et l'équation de Poisson sur l'intérieur du domaine sélectionné (résolue par technique de différences finies). Aussi, il propose quatre critères d'édition de gradients : remplacement, moyenne, somme, et maximum. Le résultat est donc $2 \times 4 = 8$ images correspondantes à tous les choix possibles.

2.1. Quel est le problème que la méthode cherche à résoudre ? Pourquoi est-ce un problème intéressant/utile/difficile ?

La méthode **Seamless Cloning**, une des méthodes de Poisson Editing, cherche à cloner certaines régions d'une image I_1 dans une image I_2 d'une manière "transparante" sans laisser des coutures visibles qui ne sont pas partiellement masquées. Ce problème est intéressant car il permet de supprimer certains objets qui sont dans l'image de départ, ou ajouter quelques régions, et mixer des zones de différentes images. La difficulté de cette méthode vient du fait que le calcul demande de résoudre un grand système linéaire sur une région du domaine d'image que l'utilisateur sélectionne manuellement. Ce qui rend le processus lent car le domaine image peut avoir une topologie complexe.

2.2. Commenter l'algorithme de *seamless cloning* (seulement sur un cas : schéma=définies finies & critère=remplacement).

- L'algorithme prend comme input l'image destination sur laquelle on souhaite effectuer une modification, et le champ vecteur $v = \Delta I_1$ obtenue à partir d'une image I_1 . L'algorithme consiste à éditer l'image I_2 en image \tilde{I}_2 dont une zone est remplacée par le domaine d'image I_1 sélectionné. Pour se faire :
- l'algorithme calcule les matrices : $\mathcal{L}, \mathcal{D}_x, \mathcal{D}_y, \mathcal{P}_\Omega, \mathcal{P}_{\partial\Omega}$ et \mathcal{S}_Ω .
- Puis il calcule la matrice $A = \mathcal{S}_\Omega \mathcal{L} \mathcal{P}_\Omega \mathcal{P}_{\partial\Omega}$ et le vecteur $b = \mathcal{S}_\Omega [\mathcal{D}_x v^x + \mathcal{D}_y v^y - \mathcal{L} \mathcal{P}_{\partial\Omega} I_2]$
- Résoudre l'équation $Ax = b$
- $\tilde{I}_2 \leftarrow I_2$: affecter le contenu d'image I_2 à l'image \tilde{I}_2
- changer le domaine $\tilde{I}_2(\Omega)$ par x .
- **Output** retourner l'image \tilde{I}_2 .

2.3. Écrire formellement la version continue de chacune des (2×4) méthodes proposées de *seamless cloning*. Plus précisément étant donné deux images $f, g : R \rightarrow \mathbf{R}$ et une région sélec-

tionnée $\Omega \subseteq R$, quelle est la définition de l'image résultante h ?

1. remplacement :

$$v = \begin{cases} \Delta f, & \text{sur } \Omega \\ \Delta g, & \text{sinon} \end{cases}$$

Ici on remplace le gradient de l'image f par celle de g sur le domaine $\Omega \subseteq R$ avec :

$$\begin{cases} \Delta h = \operatorname{div}(v) \\ h|_{\partial\Omega} = f|_{\partial\Omega} \end{cases}$$

2. Moyenne :

$$v = \frac{1}{2}(\Delta f + \Delta g)$$

Le vecteur v est la moyenne du gradient des deux images f et g .

$$\begin{cases} \Delta h = \operatorname{div}(v) \text{ sur } R \\ \partial h = \partial n \text{ sur } \partial R \end{cases}$$

3. Somme :

$$v = \Delta f + \Delta g$$

Le vecteur v est la somme du gradient de deux images f et g .

$$\begin{cases} \Delta h = \operatorname{div}(v) \text{ sur } R \\ \partial h = \partial n \text{ sur } \partial R \end{cases}$$

4. Maximum :

$$v = \begin{cases} \Delta f, & \text{si } \Delta f \geq \Delta g \\ \Delta g, & \text{sinon} \end{cases}$$

Le vecteur v est la moyenne du gradient de deux images f et g .

$$\begin{cases} \Delta h = \operatorname{div}(v) \text{ sur } R \\ \frac{\partial h}{\partial n} = 0 \text{ sur } \partial R \end{cases}$$

2.4. Pour le cas (critère=replacement), trouver un exemple de copier-coller pour lequel la méthode globale et locale donnent des résultats visuellement très différents.

La figure 2, donne l'exemple demandé.

On voit que la méthode de *Seamless Cloning* ne modifie pas l'image destination en dehors de la région Ω , donc pour avoir des résultats satisfaçants il faut sélectionner une région Ω petite, d'où une contrainte de la méthode, car la sélection doit être faite manuellement et elle prend du temps.

2.5. Pour le cas (local, différences finies), trouver un exemple de copier-coller pour lequel les critères replacement, max, et somme donnent des résultats visuellement très différents.

Voir la figure 4.

On voit que les critères replacement, max, et somme donnent des résultats visuellement très différents.

2.6. Justifier quel est le cas «idéal» d'utilisation de l'algorithme (type d'image source/destination, région à copier-coller).

Le cas idéal d'utilisation d'algorithme sera quand :

Les résultats d'algorithme sont idéaux si le gradient v d'images est assez régulier et intégrable. donc, puisque la perception visuelle de l'homme concerne que le Laplacien d'image est non l'image elle-même, l'algorithme est performant si le laplacien des deux images sont assez régulier mais il y'a d'autres petites conditions particulières sur l'image et la région à copier-coller :

- **type d'image source/destination :** L'image source doit être moins foncée que l'image destination, sinon le contrast des régions sombre sera augmenté, une chose que peut causer un contraction dans le bords de la zone Ω si le voisinage de bord $\partial\omega$ de l'image destination est plus brillant, ce qui donne des artefacts. C'est préférable que l'image source et l'image destination ont une texture assez proche, les résultats seront donc mieux car on peut éviter avoir des artefacts dans le voisinage de bord $\partial\omega$ de la région Ω à clonner.
- **region à copier-coller :** Idéalement, Ω doit avoir une topologie simple et régulière, donc si la région Ω est assez lisse, petite, convexe et ne se coupe pas avec les bords de l'image. L'algorithme fournira des résultats meilleurs si la région se situe à l'intérieur d'image destination.

2.7. Donner un exemple pour le cas décrit en (2.6.)

La figure 3 (prise du support de TP2 : <http://www.ipol.im/pub/art/2016/163/>) donne un bon exemple pour le cas décrit en (2,6). Les (deux) domaines Ω_1 et Ω_2 (voir la sélection), sont convexes et se situent à l'intérieur de l'image destination qui a un contraste plus élevé que l'image source ce qui a empêché la création des artefacts dans les bords de la région Ω .



Image source

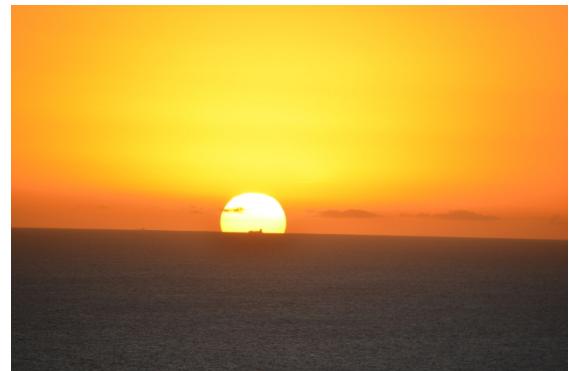


Image destination



Méthode globale



Méthode locale

FIGURE 2 – un exemple de copier-coller pour lequel la méthode globale et locale donnent des résultats visuellement très différents.

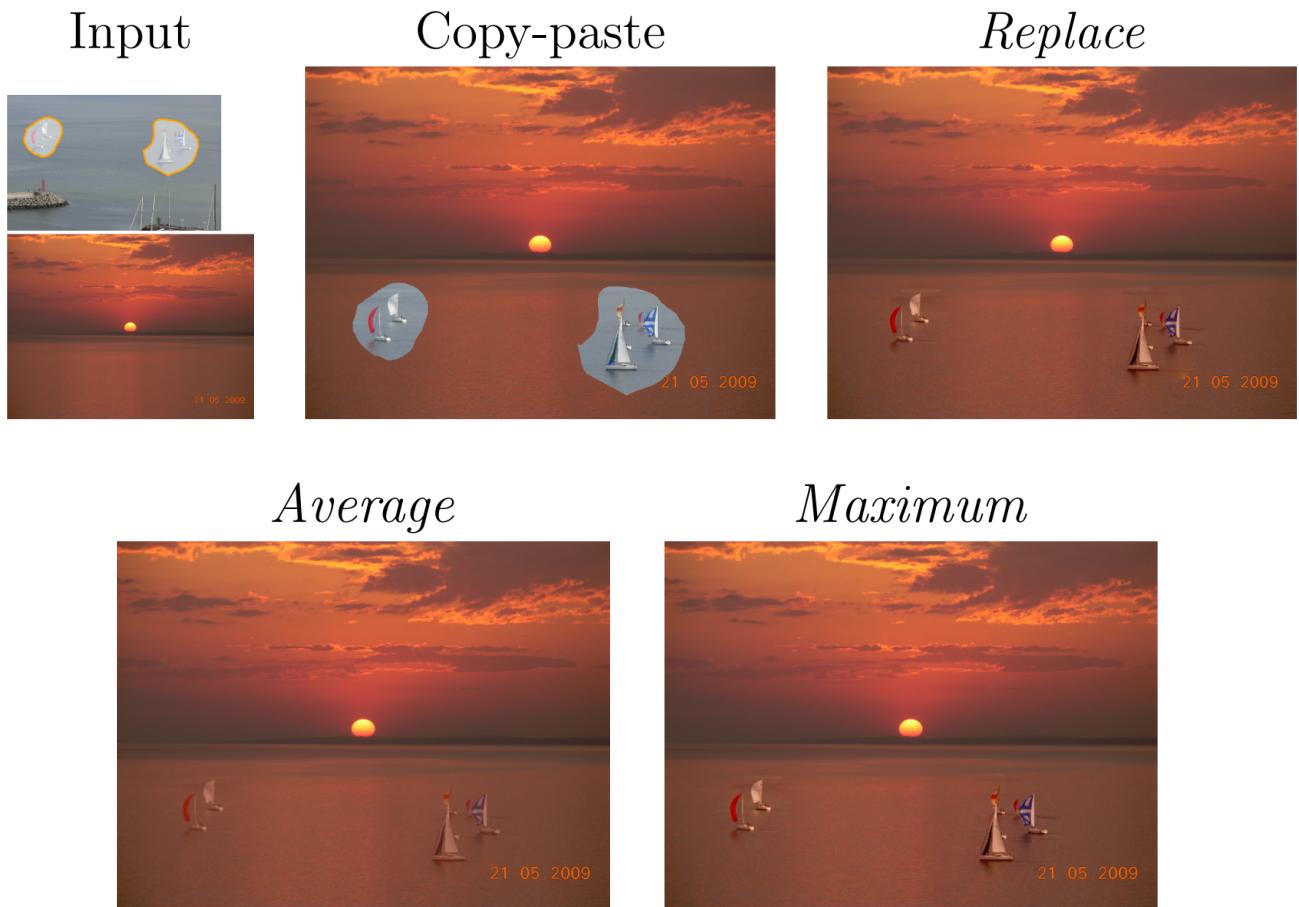


FIGURE 3 – Exemple décrivant (2.6) : finite difference numerical method.

2.8. Justifier quel est le pire cas d'utilisation, sur lequel toutes les options de l'algorithme donneraient des artefacts ou des résultats médiocres.

Le pire des cas sera certainement quand l'image source est très foncée et une image destination bien contrastée. En effet, aucune des méthodes de différences finies ne satisfait l'exigence du *seamless cloning* qui est d'éditer une image sans laisser des traces :

- **la méthode maximum** ne fournira aucune modification sur l'image destination. Car le gradient d'une image foncée est assez minime.
- **la méthode moyenne** rend la zone Ω plus grise dans l'image destination.
- **la méthode remplacement** fournit un résultat indigeste, car la zone Ω éditée n'est pas transparente comme le promis le principe de *seamless cloning*. Le bord semble flou en comparaison avec le reste de l'image destination qui est bien contrasté.
- **la méthode somme** peut fournir des résultats meilleurs que **la méthode moyenne**, mais ces résultats restent insuffisants et ne satisfont pas le principe de *Seamless Cloning*.

2.9. Donner un exemple pour le cas décrit en (2.8.)

Un exemple pour ce cas se trouve dans la figure 4.

Remarque : La sélection que j'ai fait pour la zone Ω n'est pas parfaite car je n'ai pas une souris, et il est carrément impossible de faire une sélection sans dé-zoomer, donc la zone n'est pas assez bonne mais les résultats souhaités sont claires.

****2.10.** Imaginer que vous utilisez cet algorithme (avec schéma=dérences, critère=remplacement) pour extraire un objet d'une image très foncée (planche de valeurs entre [0, 20]) et l'insérer sur une image bien contrastée (planche de valeurs entre [0, 255]), comme dans la Figure 5. Est-ce que vous attendriez un résultat satisfaisant dans ce cas ? Pourquoi ? Pensez-vous que travailler sur les images $\log \circ f$ et $\log \circ g$ changerait beaucoup le résultat ? Pourquoi ?

Les résultats sont insatisfaisants, voir la figure 4 qui est commenté en (2.8).

une différence claire entre les deux approches est que la méthode qui utilise $\log \circ f$ et $\log \circ g$ n'est pas linéaire en fonction de f et de g .

Quand l'algorithme donne son résultat, il trouve un champ gradient v gradient approprié aux fonctions $\log \circ f$ et $\log \circ g$ et qui respecte les conditions aux bords (de Neuman ou de Dirichlet). Ces conditions sont respectées par les fonctions $\log \circ f$ et $\log \circ g$, elles ne seront pas nécessairement respectées par les fonctions f et g , donc la méthode de Poisson n'est pas respectée (pour

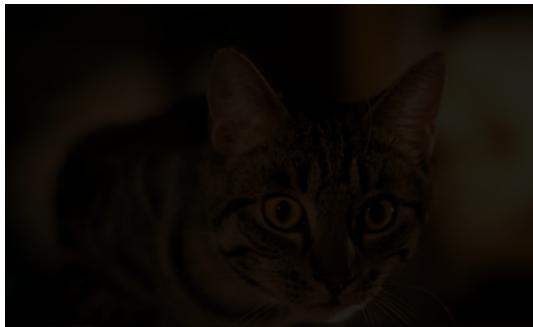
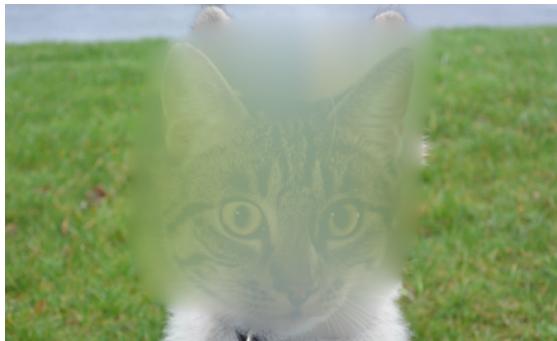


Image source = chat, Image destination= chien



Copier-Coller



Remplacement



Moyenne



Somme



Maximum

FIGURE 4 – exemple où les résultats de toutes les méthodes d'algorithme "différences finies" sont insuffisants.

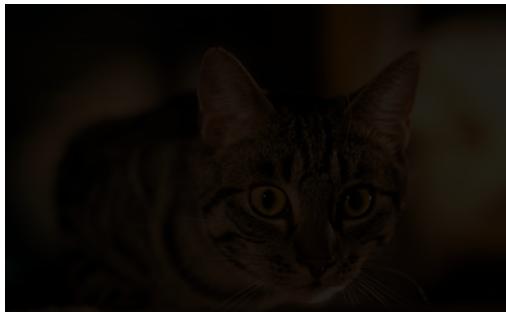


FIGURE 5 – Que se passera-t-il si on essaye de copier le visage du chat sur la tête du chien, avec l'éditeur de Poisson ?

les conditions au bord de Dirichlet). Une autre contrainte est qu'on ne peut pas trouver une fonction bijective qui nous fournit des variations (en particulier, le gradient et le laplacien) de $\log \circ f$ et $\log \circ g$ à partir des variations de fonction f et g pour reconstituer une image \tilde{f} "traitée par l'algorithme" à partir de $\log \circ f$ "qui est véritablement le résultat de l'algorithme".

Remarque : le raisonnement précédent est fait sous l'hypothèse que on veut modifier une région Ω dans l'image f , par application du Seamless cloning sur la région $\log(\Omega)$ dans l'image $\log \circ f$, puis on reconstituer la région modifiée $\tilde{\Omega}$ dans l'image destination (après sa modification) \tilde{f} .