

EQUATION DE POISSON ET TRAITEMENT D'IMAGE

GROUPE DE TRAVAIL THÉMATIQUE

4TQMS801S

Equation de Poisson et traitement d'image

Virginie MONTALIBET

Sébastien Eyzat

May 3, 2020



Contents

1 Présentation du problème	2
1.1 Problème mathématique associé	3
1.2 Résolution et équivalence avec l'équation de poisson	4
2 Résolution	5
2.0.1 Qu'est qu'une image	5
2.1 Discrétisation [4]	6
2.1.1 Notations	6
2.2 Méthode des différences finies	6
2.2.1 Résolution du système	7
2.3 Exemple	9
2.4 Fourier [4]	11
2.4.1 Rappel et définitions des opérateurs	11
2.4.2 Résolution avec la méthode Fourier	12
3 Optimisation	13
3.1 Méthode de Douglas [1]	13
3.1.1 D'un problème avec contraintes...	13
3.1.2 ... À un problème sans contraintes	13
3.1.3 La convexité...	14
3.1.4 ... Pour utiliser l'algorithme de Douglas...	14
3.1.5 ... Avec les opérateurs proximaux	15
3.1.6 Convergence de l'algorithme vers la solution	16
3.1.7 Temps et coût de l'algorithme	16
3.2 Tableau comparatif	17
4 Amélioration du collage	17
5 Résultats obtenus	18
6 Comparaison	19
6.1 Différences de résultat	20
6.2 Différence de temps	21
7 Implémentation	21
7.1 Interface	22
7.2 Organisation du code	23
8 Conclusion	25

1 Présentation du problème

Le traitement d'images est un ensemble de méthodes permettant d'étudier et de transformer une ou plusieurs images à l'aide de moyens mathématiques et numériques. Le principe du traitement d'images consiste à extraire certaines informations de celles-ci, afin de les étudier ou de les modifier. Il est utilisé dans beaucoup d'applications telles que l'amélioration du contraste, l'application d'un filtre (flou, lissage, changement de couleurs), ou encore les détections et identifications d'objets par exemple.

Dans ce rapport, nous nous intéresserons à l'incrustation d'images. À partir de deux images, comment sélectionner une partie de la première et l'incruster de la manière la plus naturelle possible dans la seconde ?

Afin d'éclaircir nos propos et d'identifier les problèmes à résoudre, voici un exemple de ce que nous souhaitons faire. A l'aide des deux images présentées ci-dessous, l'image T(arget) et l'image S(ource).



Figure 1: Image T



Figure 2: Image S

L'objectif est d'incruster toute ou partie de l'image S dans l'image T. En terme de manipulations, cela consiste à faire un mélange ou mixage des deux images, ou encore à effectuer un clonage lisse de la seconde image dans la première. Le résultat attendu pour une insertion "réussie" est un résultat comme celui présenté ci-dessous :



Figure 3: Image finale attendue

Cette image a été obtenue à l'aide d'un de nos algorithmes, vous retrouvez les différents collage à la fin de ce rapport.

Remarques Ce résultat semble naturel, les frontières entre l'image collée et l'arrière plan sont très peu visibles et ont été estompées, l'ours présent dans la première image n'a pas été déformé et semble faire parti de la photo initiale. Mais comment obtenir un tel résultat ?

En reprenant les deux images initiales, séparées, T et S et en effectuant un simple copier/coller, voici l'image que nous devrions obtenir :



Figure 4: Simple copier/coller

Ici certains pixels de T, sont écrasés par ceux de l'image qui a été ajoutée. Ce résultat n'est bien entendu pas utilisable et bien loin de l'image finale attendue. Le découpage et le collage entre l'arrière-plan et l'image "objet" sont beaucoup trop visibles, les couleurs ne sont pas les mêmes et incohérentes. Le résultat doit être beaucoup plus naturel, comme écrit plus haut. Cette simple manipulation n'est donc pas suffisante pour effectuer le clônage cohérent d'une image dans une autre.

Il semble évident de vouloir modifier l'image finale obtenue ci-dessus afin qu'elle paraisse la plus naturelle possible. Nous verrons tout au long de ce rapport, quels sont les changements à effectuer, et comment la résolution d'une équation aux dérivées partielles, l'équation de Poisson, permet d'obtenir un bien meilleur résultat. Nous implémenterons trois algorithmes permettant de résoudre le problème posé, et comparerons les résultats obtenus à l'aide de ceux-ci.

1.1 Problème mathématique associé

Comme écrit plus haut il faut apporter des modifications au précédent collage afin qu'il corresponde au mieux à nos attentes. En réalité nous ne souhaitons pas modifier l'entièreté de l'image mais seulement une "sous-image" correspondant à l'endroit du collage. Pour ce faire, considérons I, la partie de l'image finale à modifier. Il est ainsi possible de représenter le problème sous forme schématique.



Avec :

- T , l'image "Target", l'image destination, l'image sur laquelle s'effectuera le collage, l'arrière plan.
- S , l'image "Source", l'image qui sera collée.
- Ω , le domaine dans lequel se trouve l'inconnue.
- $\partial\Omega$, la frontière de Ω .
- I , l'inconnue, la partie de l'image que nous ne connaissons pas et que nous voulons trouver, elle se situe dans $\Omega \cup \partial\Omega$

Nous souhaitons donc trouver une fonction I qui satisfasse un certain nombre de critères, afin de correspondre au résultat attendu. Cette fonction représente la partie modifiée de l'image. Mais quelles sont les conditions qu'elle doit remplir pour que le rendu soit le meilleur possible? Cette nouvelle image I doit-elle être plus proche de l'image collée S , ou de l'image d'arrière-plan T ?

Pour que l'image obtenue s'incruste parfaitement, il faut que celle-ci dénature le moins possible les deux images sélectionnées au départ. En effet, les détails de l'image que nous voulons coller, S , doivent être retrouvés dans l'image finale, il ne faut donc pas modifier les variations qu'elle (S) pourrait posséder comme par exemple, les contours ou les objets lui appartenant. L'ours à coller, doit être présent dans le collage final. Il faut donc être capable de retrouver dans l'image finale, les informations présentes dans l'image initiale S . Par conséquent, il faut que les variations présentes dans I soient presque identiques à celles de S .

Rappelons qu'en traitement d'image, les variations d'une image peuvent être obtenues en calculant son gradient. En effet, une variation peut être représentée comme un changement "brutal" d'intensité entre deux pixels. Le gradient d'une image étant numériquement obtenu en effectuant la différence entre des pixels voisins : si celle-ci est élevée alors il y a un fort changement d'intensité entre eux, (par exemple un pixel noir et un autre blanc), et donc probablement la présence d'un contour. Par conséquent un gradient élevé détecte la présence de fortes variations. Au contraire, en l'absence de variations, le gradient est presque nul. Son calcul permet entre autre de détecter les contours d'une image.

Ici, il est nécessaire que les contours et objets de l'image finale soient très proches de ceux de l'image à coller. Mathématiquement, la fonction I possède donc un gradient, très proche(voir identique) à celui de l'image initiale S . Nous cherchons donc :

$$\min \iint_{\Omega} \|\nabla I_{x,y} - \nabla S_{x,y}\|^2 dx dy$$

Mais les "frontières" entre l'image collée et l'image d'arrière-plan T , ne doivent pas non plus être visibles, il faut donc que les pixels se situant sur cette partie là, i.e $\partial\Omega$, soient le plus proches possible de T . En d'autres termes, les pixels de l'image finale sur le bord, coïncident avec ceux de l'image initiale T , au même endroit. La fonction I doit donc vérifier:

$$I_{(x,y)} = T_{x,y} \text{ sur } \partial\Omega$$

Répondre au problème implique donc de résoudre un problème variationnel classique auquel des conditions sur le bord de Dirichlet sont ajoutées. Voici donc le nouveau problème à résoudre :

$$\begin{cases} \min \iint_{\Omega} \|\nabla I_{x,y} - \nabla S_{x,y}\|^2 dx dy \\ I_{(x,y)} = T_{x,y} \text{ sur } \partial\Omega \end{cases} \quad (1)$$

1.2 Résolution et équivalence avec l'équation de poisson

Pour résoudre ce problème, il faut donc trouver le minimum de la fonction g suivante :

$$g(I) = \int_{\Omega} \|\nabla I(x) - \nabla S(x)\|^2 dx \quad (2)$$

S'il existe alors celui-ci annule le gradient de la fonction.

Calcul de $\nabla g(I)$ En utilisant les formules de Taylor-Young à l'ordre 1, le gradient peut être calculé facilement :

Soit $u \in \mathbb{R}^n$, tel que $\|u\| = 1$:

$$g(I + \epsilon u) = g(I) + \epsilon \langle \nabla g(I), u \rangle + o(\epsilon)$$

Posons : $v = \nabla S$.

$$\begin{aligned} g(I + \epsilon u) - g(I) &= \int_{\Omega} \|\nabla(I + \epsilon u) - v\|^2 - \|\nabla I - v\|^2 dx \\ g(I + \epsilon u) - g(I) &= \int_{\Omega} \|\nabla I - v\|^2 + \|\nabla \epsilon u\|^2 + 2(\nabla I - v) \times \epsilon \nabla u - \|\nabla I - v\|^2 dx \\ &= \int_{\Omega} \epsilon^2 \|\nabla u\|^2 + 2(\nabla I - v) \times \epsilon \nabla u dx \\ &= 2 \int_{\Omega} (\nabla I - v) \times (\nabla \epsilon u) + O(\epsilon^2) \\ &= 2 \langle \nabla I - v, \nabla \epsilon u \rangle + O(\epsilon^2) \\ &= 2\epsilon \langle \nabla I - v, \nabla u \rangle + O(\epsilon) \\ &= 2 \langle \nabla I - v, \nabla u \rangle + O(\epsilon) \\ &= -2 \langle \operatorname{div}(\nabla I - v), u \rangle + O(\epsilon) \end{aligned}$$

Par identification, le gradient de g vaut

$$\nabla g(I) = -2(\Delta I - \operatorname{div}(v))$$

Le minimum de g annule son gradient, ainsi, si I est le minimum de la fonction alors il vérifie :

$$0 = (-\Delta I + \operatorname{div}(v))$$

$$\begin{aligned} \Delta I &= \operatorname{div}(\nabla S) \\ \Delta I &= \Delta S \end{aligned}$$

Trouver le minimum de (2) revient donc à résoudre l'équation :

$$\Delta I = \Delta S$$

Le nouveau problème est donc le suivant :

$$\begin{cases} \Delta I = \Delta S \text{ sur } \Omega \\ I = T \text{ sur } \partial\Omega \end{cases} \quad (3)$$

Il n'est autre que l'équation de Poisson. Ainsi résoudre (1), est équivalent à résoudre l'équation de poisson avec conditions aux bords de Dirichlet. Nous décrirons dans ce rapport 3 manières de résoudre numériquement cette équation.

2 Résolution

Dans cette partie nous allons résoudre (1) à l'aide de discréétisation et de différences finies. Puis nous résoudrons celle-ci à l'aide des transformées de Fourier.

2.0.1 Qu'est qu'une image

Avant de résoudre numériquement le problème, nous rappelons ce qu'est un image et comment nous la parcourrons dans la suite.

Une image peut être représentée comme une succession de pixels. En traitement d'image, ce sont d'ailleurs sur ces pixels que le traitement est effectué. Leur modification entraîne la modification de l'image globale. Il est alors possible de découper l'image, en prenant comme échelle le pixel. L'image peut donc être vue comme une grille,



Figure 5: Maillage d'une image

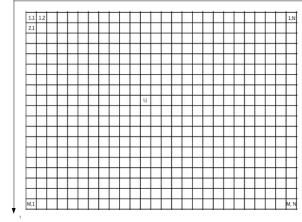


Figure 6: Parcours d'une grille

dans laquelle chaque carré représente un pixel. Les pixels seront numérotés suivant la règle suivante :

Le premier pixel est situé en haut en gauche, puis il suffit de parcourir la grille, de gauche à droite et de haut en bas, comme ci-dessus :

Les pas d'espaces sont donc égaux et valent 1. Dans la suite nous considérerons que l'image à modifier est de taille $M \times N$.

2.1 Discréétisation [4]

Cette partie sera consacrée à la résolution du problème à l'aide des différences finies. Pour trouver la solution il faut donc discréétiser les Lapaciens des images, puis résoudre un système.

2.1.1 Notations

Le gradient:

Le gradient est un vecteur composé des dérivées partielles d'une fonction. Soit la fonction, $f(x,y)$, on note le gradient de f :

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$$

Le Laplacien

On note le Laplacien : Δ , et $\Delta = \operatorname{div}(\nabla f)$.

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

2.2 Méthode des différences finies

Le problème à résoudre est le suivant :

$$\begin{cases} \Delta I = \Delta S \text{ sur } \Omega \\ I = T \text{ sur } \partial\Omega \end{cases}$$

Dans un premier temps, discréétisons le Laplacien de I . Le Laplacien étant la somme des dérivées partielles secondes : $\frac{\partial^2 I}{\partial x^2}$ et $\frac{\partial^2 I}{\partial y^2}$, sa discréétisation commence par une discréétisation de celles-ci. En utilisant les formules de Taylor-Young à l'ordre 2 suivantes :

$$I(x+h, y) = I(x, y) + h \times \frac{\partial I(x, y)}{\partial x} + \frac{h^2}{2} \times \frac{\partial^2 I(x, y)}{\partial x^2} + o(h^3)$$

$$I(x-h, y) = I(x, y) - h \times \frac{\partial I(x, y)}{\partial x} + \frac{h^2}{2} \times \frac{\partial^2 I(x, y)}{\partial x^2} + o(h^3)$$

Il est donc facile de voir que la somme de ces deux équations permet d'obtenir une discréétisation de la dérivée seconde : $\frac{\partial^2 I(x, y)}{\partial x^2}$:

$$\frac{\partial^2 I(x, y)}{\partial x^2} = \frac{1}{h^2} (I(x+h, y) + I(x-h, y) - 2 \times I(x, y))$$

La somme des discrétisations des dérivées secondes : $\frac{\partial^2 I(x,y)}{\partial x^2}$ et $\frac{\partial^2 I(x,y)}{\partial y^2}$, permet d'obtenir une discrétisation possible du Laplacien de I : ΔI .

$$\Delta I(x, y) = \frac{I(x+h, y) + I(x-h, y) - 2 \times I(x, y)}{h^2} + \frac{I(x, y+k) + I(x, y-k) - 2 \times I(x, y)}{k^2}$$

Les pas d'espaces h et k étant égaux à 1, nous pouvons écrire une discrétisation du Laplacien :

$$\Delta I(x, y) = I(x+1, y) + I(x-1, y) + I(x, y+1) + I(x, y-1) - 4 \times I(x, y)$$

Application à une image Afin d'obtenir le Laplacien du pixel $I(i,j)$, il est donc nécessaire d'avoir la connaissance de ses pixels voisins que nous nommerons par la suite $U(p)$, $D(own)$, $L(eft)$, $R(ight)$ pour les pixels $I(i-1,j)$, $I(i+1,j)$, $I(i,j-1)$, $I(i,j+1)$.

	1	
1	-4	1
	1	

Figure 7: Laplacien du pixel

Afin de trouver la solution au problème, il faut donc appliquer cette discrétisation à chaque pixel de la partie de l'image à modifier($\Omega \cup \partial\Omega$). Chaque pixel faisant intervenir ses voisins, la résolution du problème passe donc par la résolution d'un système. En notant

$$g(x, y) = S(x+1, y) + S(x-1, y) + S(x, y+1) + S(x, y-1) - 4 \times S(x, y)$$

Résoudre $\Delta I(x, y) = \Delta S(x, y)$ sur Ω est équivalent à résoudre :

$$\begin{cases} I(i+1, j) + I(i-1, j) + I(i, j+1) + I(i, j-1) - 4 \times I(i, j) = g(i, j) \\ \text{pour } (i, j) \in \Omega \\ I(i, j) = T(i, j) \text{ pour } (i, j) \in \partial\Omega \end{cases}$$

La résolution de ce système de taille $M \times N$, permettra de trouver la nouvelle image I , et donc de résoudre numériquement l'équation de Poisson avec conditions aux bords de Dirichlet. Voici, le système obtenu :

$$\left\{ \begin{array}{l} I(1, 1) = T(1, 1) \\ I(3, 2) + I(1, 2) + I(2, 3) + I(2, 1) - 4I(2, 2) = g(2, 2) \\ I(3, 3) + I(1, 3) + I(2, 4) + I(2, 2) - 4I(2, 3) = g(2, 3) \\ \dots \\ I(M, N-1) + I(M-2, N-1) + I(M-1, N) + I(M-1, N-2) - 4I(M-1, N-1) = g(M-1, N-1) \\ I(M, N) = T(M, N) \end{array} \right. \quad (4)$$

2.2.1 Résolution du système

Afin de résoudre ce système, il est plus facile de l'écrire sous forme matricielle. Il faut maintenant trouver la solution du problème suivant :

$$AI = b$$

Si la matrice est inversible, alors la solution est évidente, et elle vaut :

$$I = A^{-1} \times b$$

Avec :

- A, une matrice carrée de taille ($M \times N, M \times N$)
- I, un vecteur colonne de taille ($M \times N, 1$)
- b, un vecteur colonne de taille ($M \times N, 1$)

Voici donc à quoi ressemble le système que nous souhaitons résoudre : La matrice A, est une matrice par bloc. Pour remplir la matrice, correspondante : commençons par écrire l'image sous forme d'une vecteur colonne. Dans l'exemple ci-dessous nous voyons bien qu'afin de remplir les lignes de la matrice correspondant au pixels se situant à l'intérieur du domaine, il nous faut la connaissances des voisins. Ainsi dans l'exemple précédent, le calcul du

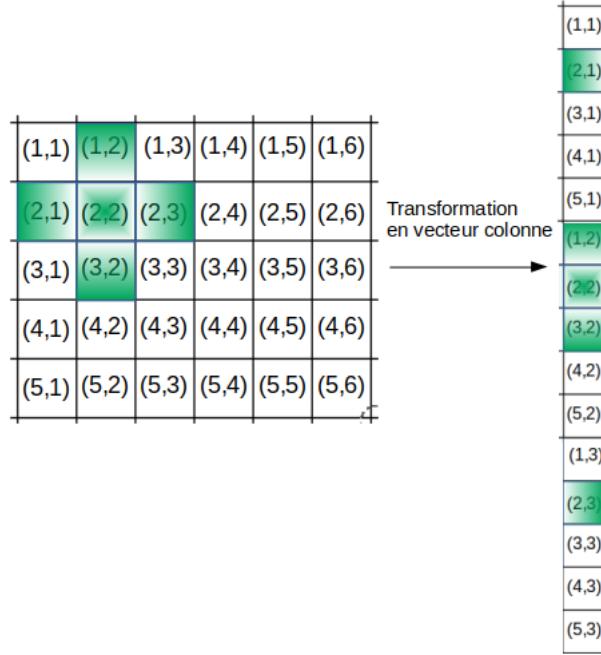


Figure 8: Transformation en un vecteur colonne

Laplaciens du pixel (2,2) nécessite les pixels U,D,R,L. Il faut donc qu'ils soient présents dans la matrice A. Grâce au vecteur colonne obtenu, il est évident que la ligne de la matrice correspondant au pixel à l'intérieur du domaine sera remplie de la manière suivante :

$$(1 \ 0 \ \dots \ 1 \ -4 \ 1 \ 0 \ \dots \ 1 \ 0 \dots) \quad (5)$$

Si l'image est de taille ($M \times N$) alors :

- Le coefficient (i, j-1) dans la matrice correspondant au pixel U de l'image
- Le coefficient (i,j+1)dans correspondant au pixel D
- Le coefficient (i,j+M) correspondant au pixel R
- Le coefficient (i,j-M) correspondant au pixel L

Les blocs de la matrice, correspondant aux pixels situés sur le bords du domaine seront remplis de la façon suivante :

$$\begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 0 & 0 \dots & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & \dots \\ \dots & & & & & & & & & \\ 0 & 0 & \dots & 0 & 0 & 0 & 0 \dots & 0 & 0 & 1 \end{pmatrix} \quad (6)$$

Le vecteur b , contient les Laplaciens des pixels à l'intérieur de Ω et les valeurs des pixels de T , en dehors du domaine. Les valeurs des pixels de celle-ci, étant connus, il est facile de calculer son Laplacien en utilisant les discrétilisations vues ci-dessus. Ainsi :

$$b(i) = \begin{cases} T(i, j) & \text{si } I(i, j) \notin \Omega \\ g(i, j) & \text{si } I(i, j) \in \Omega \end{cases}$$

En inversant la matrice, la solution I obtenue est un vecteur :

$$\begin{pmatrix} I(1, 1) \\ I(2, 1) \\ \dots \\ I(1, 2) \\ I(2, 2) \\ \dots \\ I(M, N) \end{pmatrix}$$

Afin de trouver l'image finale, il est donc important de reconstruire une matrice de taille $M \times N$, à partir de ce vecteur.

2.3 Exemple

Considérons l'image S ci-contre que nous souhaitons coller. Nous souhaitons ici coller les deux carrés rouge sur une

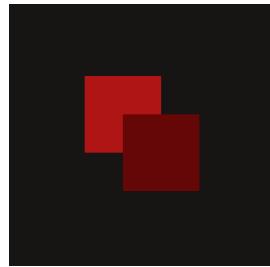


Figure 9: Images à coller

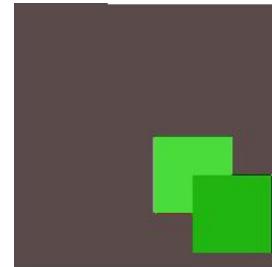


Figure 10: Image de fond

image que nous nommerons T .

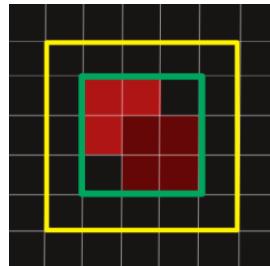


Figure 11: Sélection à coller

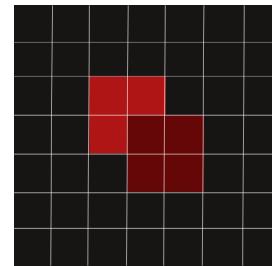


Figure 12: Vue grille pixel

Ici, notons Ω , l'intérieur de la zone encadrée par la ligne verte, et des $\partial\Omega$, les pixels appartenant à $Jaune \setminus \Omega$

Construction du système

$$\left\{ \begin{array}{l} I_{1,1} = T_{1,1} \\ \dots \\ I_{5,1} = T_{5,1} \\ I_{1,2} = T_{1,2} \\ \Delta I_{2,2} = \Delta S_{2,2} \\ \Delta I_{3,2} = \Delta S_{3,2} \\ \Delta I_{4,2} = \Delta S_{4,2} \\ \dots \end{array} \right. \quad (7)$$

Avec $\Delta I(i, j) = U + D + L - 4I(i, j)$.

En écrivant ce système sous forme matricielle la matrice A est la matrice 25×25 ci-dessous :

Figure 13: Matrice du système

$$I = \begin{pmatrix} I(1,1) \\ I(2,1) \\ \dots \\ I(5,1) \\ \dots \\ I(1,3) \\ \dots \\ I(5,3) \\ \dots \\ I(5,5) \end{pmatrix} \quad b = \begin{pmatrix} T(1,1) \\ \dots \\ \Delta S(2,2) \\ \dots \\ T(5,2) \\ \Delta S(1,3) \\ \dots \\ T(5,3) \\ \Delta S(2,4) \\ \dots \\ \Delta S(4,4) \\ T(5,4) \\ \dots \\ T(5,5) \end{pmatrix} \quad (8)$$

La solution de ce système existe bien. En effet, A est carrée, elle est de taille $(M \times N, M \times N)$. Elle est aussi, toujours remplie de la manière, 'par blocs'. Ses colonnes étant linéairement indépendantes, elle est donc inversible.

La solution I, s'écrit sous la forme

$$I = A^{-1}b.$$

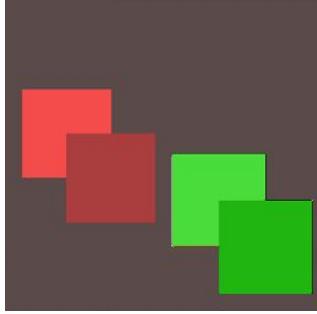


Figure 14: Résultat obtenu

Nous ajouterons dans la section suivante les résultats obtenus à l'aide de cette méthode.

Celle-ci fonctionne très bien mais le temps de calcul peut devenir très long. En effet, cette méthode demande une inversion matricielle, et donc un temps de calcul relativement long sur de "très" grands systèmes, donc de très grandes sélections. Nous allons maintenant voir une seconde méthode, plus rapide, en nous plaçant dans le domaine de Fourier.

2.4 Fourier [4]

Avec cette seconde méthode nous allons résoudre l'équation de Poisson à l'aide de la transformée de Fourier. Avant de formuler la résolution de ce problème. Rappelons la définition des opérateurs dont nous aurons besoin dans la suite. [5]

2.4.1 Rappel et définitions des opérateurs

Transformée de Fourier (discrète) Soit F une fonction, sa transformée de Fourier peut s'écrire de la façon suivante :

$$\hat{F}(x, y) = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} F(k, l) e^{-2\pi i (\frac{k \times x}{M} + \frac{l \times y}{N})} \quad (9)$$

Enfin afin de retrouver la fonction initiale nous aurons besoin de la transformée de Fourier inverse :

$$F(k, l) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \hat{F}(x, y) e^{2\pi i (\frac{xk}{M} + \frac{yl}{N})} \quad (10)$$

Gradient Pour résoudre le problème nous avons besoin de calculer les Laplaciens des images. Mais avec cette méthode nous nous placerons dans le domaine de Fourier, il est donc nécessaire de calculer la transformée de Fourier du Laplacien d'une fonction , et donc le gradient de celle-ci. F est toujours la fonction que nous souhaitons étudier.

$$\widehat{\nabla(F)} = \left(\begin{array}{c} \widehat{\frac{\partial F}{\partial k}} \\ \widehat{\frac{\partial F}{\partial l}} \end{array} \right) \quad (11)$$

En dérivant l'expression ci-dessus par rapport à la première variable :

$$\begin{aligned} \frac{\partial F}{\partial k} &= \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \hat{F}(x, y) e^{2\pi i (\frac{k \times x}{M} + \frac{l \times y}{N})} \left(\frac{2\pi ix}{M} \right) \\ &= \left(\frac{2\pi ix}{M} \right) \hat{F}(k, l) \\ \widehat{\frac{\partial F}{\partial k}} &= \left(\frac{2\pi ix}{M} \right) \widehat{\hat{F}(k, l)} \end{aligned} \quad (12)$$

Le calcul est similaire pour $\widehat{\frac{\partial F}{\partial l}}$.

On a donc :

$$\begin{aligned}\widehat{\frac{\partial F}{\partial k}} &= \left(\frac{2\pi i}{M}x\right)\widehat{F} \\ \widehat{\frac{\partial F}{\partial l}} &= \left(\frac{2\pi i}{N}y\right)\widehat{F}\end{aligned}\tag{13}$$

Laplacien

$$\begin{aligned}\frac{\partial^2 F}{\partial k^2} &= \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \widehat{F}(x, y) e^{2\pi i \left(\frac{k \times x}{M} + \frac{l \times y}{N}\right)} \left(\frac{2\pi i x}{M}\right)^2 \\ &= \left(\frac{2\pi i x}{M}\right)^2 F(k, l) \\ \widehat{\frac{\partial^2 F}{\partial k^2}} &= \left(\frac{2\pi i x}{M}\right)^2 \widehat{F(k, l)}\end{aligned}\tag{14}$$

On a donc : $\widehat{\Delta F} = \widehat{\frac{\partial^2 F}{\partial k^2}} + \widehat{\frac{\partial^2 F}{\partial l^2}}$.

$$\widehat{\Delta F} = \left(\frac{2\pi i x}{M}\right)^2 \widehat{F} + \left(\frac{2\pi i y}{N}\right)^2 \widehat{F}\tag{15}$$

2.4.2 Résolution avec la méthode Fourier

La résolution dans le domaine de Fourier, nécessite quelques changements. Cette méthode ne fonctionnant que sur un domaine rectangulaire, nous devons donc modifier le domaine Ω .

Rappelons que nous voulons résoudre le problème suivant :

$$\begin{cases} \Delta I(x, y) = \Delta S(x, y) \text{ si } (x, y) \in \Omega \\ I(x, y) = T(x, y) \text{ si } (x, y) \notin \Omega \end{cases}\tag{16}$$

Afin d'obtenir un domaine régulier considérons R , le domaine de l'image entière.

Nous voulions que le laplacien de I à l'intérieur de Ω corresponde à celui de S . En considérant le nouveau domaine Ω_2 comme étant l'image entière, il faudrait donc que le laplacien de I , soit très proche du laplacien de $T \cup S$. Mais dans ce cas, nous aurions un fort gradient sur $\partial\Omega$, (le changement d'intensité entre T et S étant fort). Il faut donc reconsidérer le problème précédent.

La nouvelle hypothèse que nous pouvons faire est donc la suivante : ∇I doit être très proche de ∇S dans Ω mais aussi très proche de ∇T dans $\Omega_2 \setminus \Omega$. En notant V :

$$V = \begin{cases} \nabla S(x, y) \text{ si } (x, y) \in \Omega \\ \nabla T(x, y) \text{ si } (x, y) \notin \Omega \end{cases} = \begin{pmatrix} V_1 \\ V_2 \end{pmatrix}\tag{17}$$

Les deux images suivantes ont été obtenues à l'aide d'un algorithme codé en python.

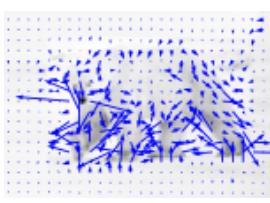


Figure 15: Champs de vecteurs de l'image S

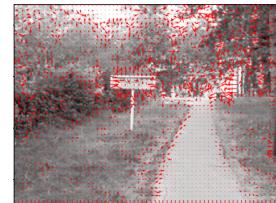


Figure 16: Champs de vecteur de l'image T

Nous devons donc résoudre l'équation suivante :

$$\Delta I = \operatorname{div}(V)$$

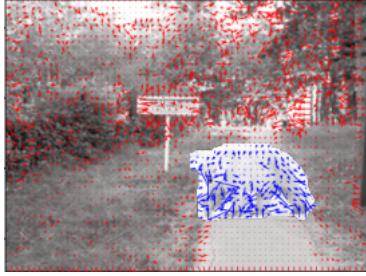


Figure 17: Nouveau domaine

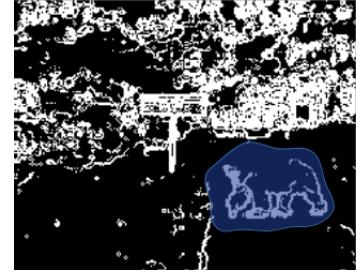


Figure 18: Nouveau domaine

Afin de pouvoir résoudre cette équation il faut imposer des conditions sur le bord. En appliquant l'effet miroir à l'image initiale alors, on obtient un signal symétrique et on peut appliquer la transformée de Fourier, pour résoudre le problème. Les conditions imposées sont donc des conditions de Neumann.

Ainsi en calculant la transformée de Fourier du laplacien, on obtient : $\widehat{\Delta I} = \widehat{\operatorname{div}(V)}$

$$\begin{aligned} \left(\frac{2\pi ix}{M}\right)^2 \widehat{I} + \left(\frac{2\pi iy}{N}\right)^2 \widehat{I} &= \left(\frac{2\pi ix}{M}\right) \widehat{V}_1 + \left(\frac{2\pi iy}{N}\right) \widehat{V}_2 \\ \left(\left(\frac{2\pi ix}{M}\right)^2 + \left(\frac{2\pi iy}{N}\right)^2\right) \widehat{I} &= \left(\frac{2\pi ix}{M}\right) \widehat{V}_1 + \left(\frac{2\pi iy}{N}\right) \widehat{V}_2 \end{aligned} \quad (18)$$

$$\widehat{I} = \frac{\left(\frac{2\pi ix}{M}\right) \widehat{V}_2 + \left(\frac{2\pi iy}{N}\right) \widehat{V}_1}{\left(\left(\frac{2\pi ix}{M}\right)^2 + \left(\frac{2\pi iy}{N}\right)^2\right)} \quad (19)$$

Afin de retrouver I , il suffit d'appliquer la transformée inverse, à l'équation ci-dessus. Les résultats obtenus avec cette méthode seront présentés dans une prochaine section.

3 Optimisation

3.1 Méthode de Douglas [1]

Nous avons vu deux manières de résoudre l'équation de Poisson avec conditions aux bords de Dirichlet. Nous allons maintenant présenter une troisième méthode :

Le méthode de Douglas

Soit I , l'image à retrouver :

3.1.1 D'un problème avec contraintes...

Remarquons que le problème initial est un problème d'optimisation avec contraintes. En effet, nous voulons minimiser $\int_{\Omega} ||\nabla I - \nabla S||^2$. Avec la contrainte suivante : $I = T$ en dehors du domaine.

Ce problème d'optimisation peut donc être résolu à l'aide de différents algorithmes, mais avant ça, transformons-le en un problème sans contraintes.

3.1.2 ... À un problème sans contraintes

En utilisant des fonctions de pénalisation nous remarquons aisément que ce problème peut être ramené à un problème sans contraintes. Réécrivons donc celui-ci

$$\min \int_{\Omega} ||\nabla I - \nabla S||^2 + \mathbb{1}_{D \setminus \Omega}(I)$$

avec

$$\mathbb{1}_{D \setminus \Omega}(I) = \begin{cases} 0 & \text{si } I \in T \setminus \Omega \\ +\infty & \text{sinon} \end{cases}$$

Dans la suite nous noterons $K = T \setminus \Omega$. K représente donc l'ensemble des images dont les pixels situés en dehors du domaine Ω , coïncident avec T .



Figure 19: Exemple d'images appartenant à K

Nous avons bien équivalence entre notre problème sans contraintes et le problème (1). En effet, si $I \in K$, alors l'indicatrice vaut 0 et nous devons juste résoudre $\min \int_{\Omega} \|\nabla I - \nabla S\|^2$. Si au contraire $I \notin K$, alors nous devons minimiser quelque chose qui vaut plus $+\infty$. Le minimum n'existe pas, il n'y a pas de solutions.

En effet, l'image I ne coïncide pas avec T à l'extérieur de Ω , la condition $I = T$ en dehors du domaine n'étant pas respectée, le problème n'a pas de solution.

Ce problème sans contraintes, traduit bien celui avec contraintes. Nous pouvons donc essayer de résoudre celui-ci, numériquement. Afin d'être sûrs que le minimum existe, nous montrerons dans la suite que cette fonction est bien convexe. Par commodité, nous noterons dans la suite :

$$\begin{aligned} F(I) &= \int_{\Omega} \|\nabla I - \nabla S\|^2 + \mathbb{1}_{T \setminus \Omega}(I) \\ &= f(I) + g(I) \end{aligned}$$

3.1.3 La convexité...

Montrons que K est convexe. Soit u et v deux images appartenant à K , alors, les pixels de u et de v , se situant à l'extérieur de Ω , coïncident avec les pixels de T .

Considérons maintenant une nouvelle image :

$$M = \lambda u + (1 - \lambda)v$$

Les pixels de u et v coïncidant avec ceux de T à l'extérieur, nous pouvons réécrire les pixels de M de la manière suivante.

$$M(i, j) = \begin{cases} \lambda u(i, j) + (1 - \lambda)v(i, j), & (i, j) \in \Omega \\ \lambda T(i, j) + (1 - \lambda)T(i, j) & (i, j) \notin \Omega \end{cases}$$

Ainsi, pour $(i, j) \notin \Omega$:

$$M_{i,j} = \lambda T_{i,j} + (1 - \lambda)T_{i,j} = T_{i,j}$$

Ainsi, les pixels de M n'appartenant pas à Ω , coïncident avec T . M appartient bien à K . Et K est donc convexe. K étant convexe, et non vide, (T en particulier appartient à K), alors la fonction $\mathbb{1}_K(I)$ est convexe. Enfin montrons la convexité de $\|\nabla I - \nabla S\|^2$. La norme étant une fonction convexe et croissante, alors la fonction : $\|\cdot\|^2$ est elle aussi convexe. Nous avons donc f et g , convexes, ainsi, la fonction $F = f + g$ est elle aussi convexe. Elle admet donc un minimum. Nous pouvons ainsi résoudre numériquement ce problème.

3.1.4 ... Pour utiliser l'algorithme de Douglas...

L'algorithme que nous allons utiliser est l'algorithme de Douglas-Rachford. Cet algorithme permet d'approcher le minimum d'une fonction $F = f + g$, f et g étant des fonctions convexes, comme montré dans la partie précédente, nous pouvons utiliser cet algorithme.

L'algorithme À chaque itération, sont calculés :

$$\begin{aligned}x_{k+1} &= prox_f(y_k) \\y_{k+1} &= y_k + prox_g(2x_{k+1} - y_k) - x_{k+1}\end{aligned}$$

Il est donc nécessaire de calculer les opérateurs proximaux respectifs de f et g .

3.1.5 ... Avec les opérateurs proximaux ...

Un opérateur proximal est défini comme suit :

$$prox_f(x) = argmin_u \left\{ \frac{\|u - x\|^2}{2} + f(u) \right\}$$

[3]

Opérateur proximal de f

$$prox_f(x) = argmin_u \left\{ \frac{\|u - x\|^2}{2} + \|\nabla u - \nabla S\|^2 \right\}$$

Afin de faciliter les notations notons :

$$h(u) = \frac{\|u - x\|^2}{2} + \|\nabla u - \nabla S\|^2$$

Nous cherchons donc

$$argmin_u h(u)$$

ie. u qui minimise la fonction h , autrement dit, un u qui annule le gradient de h .

En utilisant Taylor Young,

$$\begin{aligned}h(u + k) - h(u) &= \frac{\|u + k - x\|^2}{2} + \|\nabla(u + k) - \nabla S\|^2 - \frac{\|u - x\|^2}{2} - \|\nabla u - \nabla S\|^2 \\&= \frac{\|k\|^2 + 2\langle u - x, k \rangle}{2} + \|\nabla k\|^2 + 2\langle \nabla u - \nabla S, \nabla k \rangle \\&= O(\|k\|^2) + \langle u - x, k \rangle - 2\langle \nabla u - \nabla S, k \rangle \\&= \langle u - x - 2\text{div}(\nabla u - \nabla S), k \rangle\end{aligned}$$

Nous obtenons le gradient de h .

$$\begin{aligned}\nabla h(u) &= u - x - 2\text{div}(\nabla u - \nabla S) \\&= u - x - 2(\Delta u - \Delta S)\end{aligned}$$

En résolvant $\nabla h(u) = 0$, nous pourrons trouver : $prox_f(x)$.

$$\begin{aligned}\nabla h(u) &= 0 \\u - x - 2(\Delta u - \Delta S) &= 0 \\u - 2\Delta u &= x - 2\Delta S\end{aligned}$$

Afin de trouver u , nous utiliserons la méthode des différences finies. En discréétisant le laplacien de u , comme nous l'avons vu dans la section (1) :

$$-2u(x+1, y) - 2u(x-1, y) - 2u(x, y+1) - 2u(x, y-1) + 9 \times u(x, y) = y_k - 2\Delta S(x, y)$$

En mettant ce système sous forme matricielle nous pourrons approcher u en faisant une inversion matricielle. Nous pouvons donc numériquement approcher $prox_f(x)$.

Opérateur proximal de g g étant la fonction indicatrice suivante :

$$\mathbb{1}_{D \setminus \Omega}(I) = \begin{cases} 0 & \text{si } I \in T \setminus \Omega \\ +\infty & \text{sinon} \end{cases}$$

Nous avons donc

$$prox_g(x) = argmin_u \left\{ \frac{\|u - x\|^2}{2} + \mathbb{1}_K(u) \right\}$$

Nous savons que $prox_g(x)$ existe puisque la fonction g est convexe et la fonction norme est elle aussi convexe. Notons $h(u) = \frac{\|u - x\|^2}{2} + \mathbb{1}_K(u)$. Comme nous l'avons cette fonction n'admet un minimum que si $u \in K$. Supposons donc $u \in K$. Alors chercher $argmin_u \{h(u)\}$ est équivalent à chercher $argmin_{u \in K} \left\{ \frac{\|u - x\|^2}{2} \right\}$.

Dans notre cas, u $prox_g(x) = L$. L étant une image appartenant à K , et dont les pixels à l'intérieur de Ω coïncident avec x .

3.1.6 Convergence de l'algorithme vers la solution

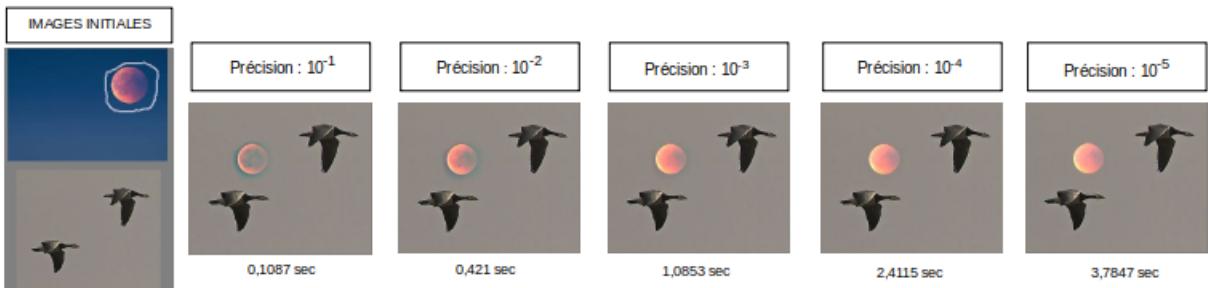


Figure 20: Convergence de l'algorithme vers la solution

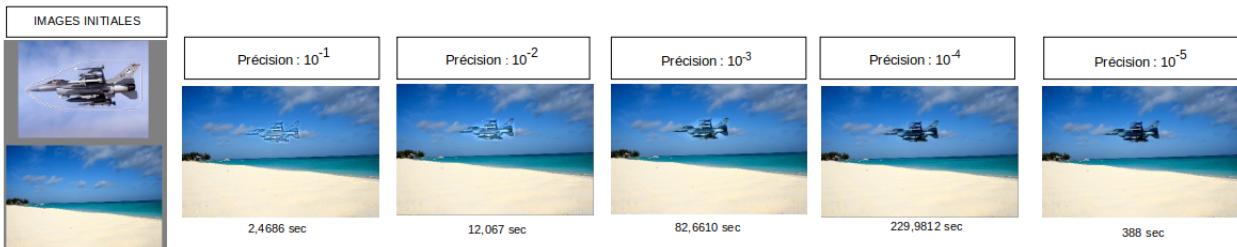


Figure 21: Convergence de l'algorithme vers la solution

L'algorithme de Douglas converge bien vers une solution I qui résout le problème initial.

3.1.7 Temps et coût de l'algorithme

L'algorithme semble être très efficace, malheureusement en termes de temps de calcul, il est très long. Nous ferons dans la prochaine partie un tableau comparatif des temps de calcul effectués sur des images de tailles différentes. Nous avons considérablement amélioré le temps de calcul de l'algorithme de Douglas-Rachford, en utilisant l'algorithme du gradient conjugué [2], afin d'inverser la matrice de l'opérateur proximal de f . Malgré tout, l'algorithme est encore lent sur de grandes images. Nous choisissons une précision de 10^{-5} , afin d'avoir des résultats convenables et comparables à ceux obtenus avec les méthodes précédentes. Le nombre d'itérations de l'algorithme devient alors très important. Pour certaines images (voir ci-dessous) une précision de 10^{-3} de l'algorithme est suffisante pour obtenir des résultats convenables. Pour d'autres en revanche il faut augmenter cette précision (voir ci-dessus avec

l'avion).

Nous avons amélioré les temps de calculs en appliquant une méthode de sur-relaxation sur l'algorithme de Douglas-Rachford, avec un $\rho = 1.85$. A chaque itération nous calculons donc :

- $x_{k+1} = prox_f(y_k)$
- $y_{k+1} = y_k + \rho_k(prox_g(2x_{k+1} - y_k) - x_{k+1})$

$(1 < \rho < 2)$.

Cependant, le temps de calcul est encore élevé sur de grandes images. Afin de réduire ce temps de calcul il faudrait donc que l'algorithme converge encore plus rapidement vers la solution. Pour cela, nous proposons d'augmenter l'ordre de discréétisation du Laplacien. Nous étions jusqu'à présent d'ordre 2, peut-être faudrait-il augmenter cette précision jusqu'à l'ordre 8.

3.2 Tableau comparatif

Taille de l'image S	Taille de l'image T	Temps Douglas	Temps Douglas (R)	Temps différences finies	Temps Fourier
$220 \times 154 \times 3$ px	$304 \times 252 \times 3$ px	834 itérations précision : 5×10^{-5} 3.799 secondes	547 itérations précision : 5×10^{-5} 2.2233 secondes	0.335 secondes	0.355 secondes
$304 \times 252 \times 3$ px	$770 \times 844 \times 3$ px	1591 itérations précision : 5×10^{-5} 72.348 secondes	1182 itérations précision : 5×10^{-5} 46.1067 secondes	0.797 secondes	1.640 secondes
$400 \times 300 \times 3$ px	$1200 \times 800 \times 3$ px	6224 itérations précision : 5×10^{-5} 440.886 secondes	3884 itérations précision : 5×10^{-5} 205.8105 secondes	1.05 secondes	2.102 secondes
$400 \times 300 \times 3$ px	$614 \times 441 \times 3$ px	5448 itérations précision : 5×10^{-5} 366.0.34 secondes	4046 itérations précision : 5×10^{-5} 224.3881 secondes	0.877 secondes	0.8576 secondes
$220 \times 154 \times 3$ px	$263 \times 192 \times 3$ px	861 itérations précision : 5×10^{-5} 4.056 secondes	510 itérations précision : 5×10^{-5} 2.3466 secondes	0.0892 secondes	0.1754 secondes

Temps Douglas (R) est la méthode de surrelaxation couplée à la méthode de Douglas, $\rho = 1.89$. Nous remarquons qu'à l'aide de cette méthode, nous pouvons considérablement diminuer le nombre d'itérations de l'algorithme de Douglas-Rachford (il converge plus vite) et ainsi améliorer le temps de calcul de celui-ci. (jusqu'à t/2, sur de grandes images). Les résultats que nous obtenons avec les différentes images sont très similaires à ceux obtenus avec les différences finies. Nous pouvons bien sûr raccourcir le temps de calcul de l'algorithme de Douglas-Rachford, en diminuant la précision. Mais dans ce cas, certains résultats sont moins convenables.

4 Amélioration du collage

Après avoir implémenté des "sliders", permettant de bouger la sélection sur l'image de fond, nous nous sommes compte que la sélection est parfois trop grande, et peut cacher des objets, ou choses importantes sur l'image de fond : Nous voyons bien, ci-dessus, qu'en sélectionnant une grande zone autour de la lune dans l'image S, le collage cache effectivement l'oiseau, objet pourtant important de l'image T. Afin de résoudre ce problème, nous avons opté pour la comparaison des gradients dans chaque image. En effet, si le gradient de T est supérieur à celui de S, alors cela signifie que l'image T, possède un objet, ou un contour à cette position tandis que S ne possède rien d'important à la même position. Nous pouvons donc "réduire" la sélection et la modifier en ce pixel. Nous obtenons donc la figure suivante en utilisant cette propriété :

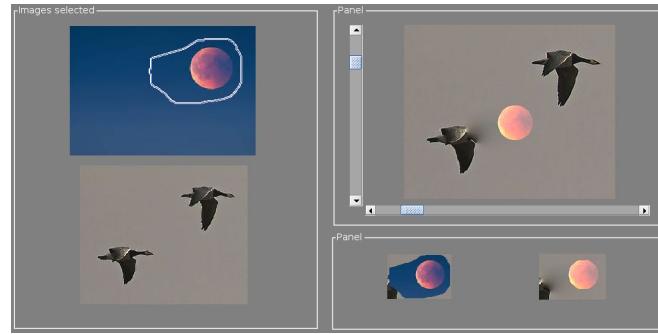


Figure 22: Problème rencontré



Figure 23: Problème résolu, algorithme utilisé : Différence Finie

Cependant cette "amélioration" à ses limites, en effet, l'intérieur de la lune ne possède que très peu de variations et presque aucun contours, ainsi, si la lune se situait exactement sur la tête de l'oiseau, certains pixels à l'intérieur de celle-ci serait "supprimé" de la sélection, et ainsi, le résultat ne serait pas satisfaisant.

5 Résultats obtenus

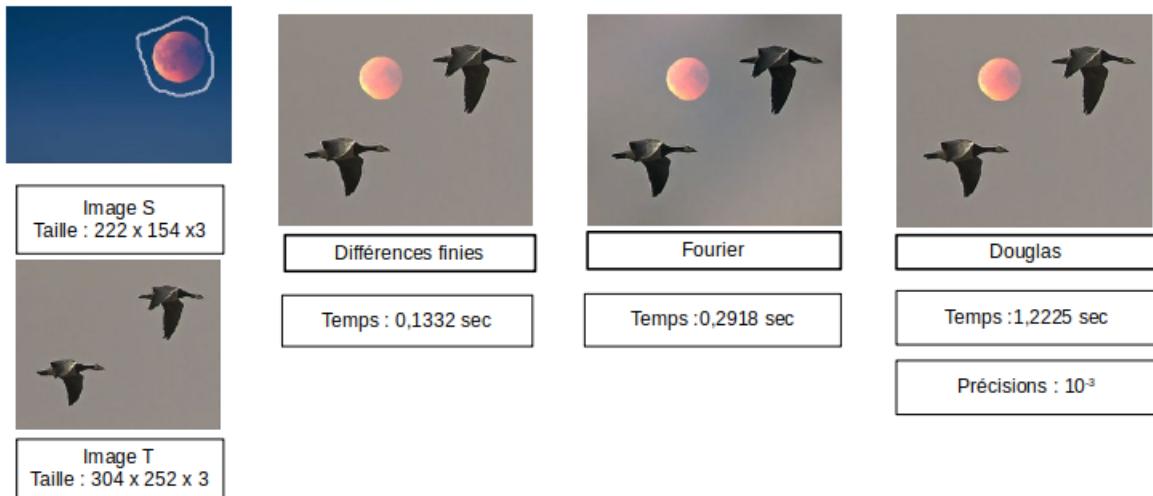


Figure 24: Résultats obtenus

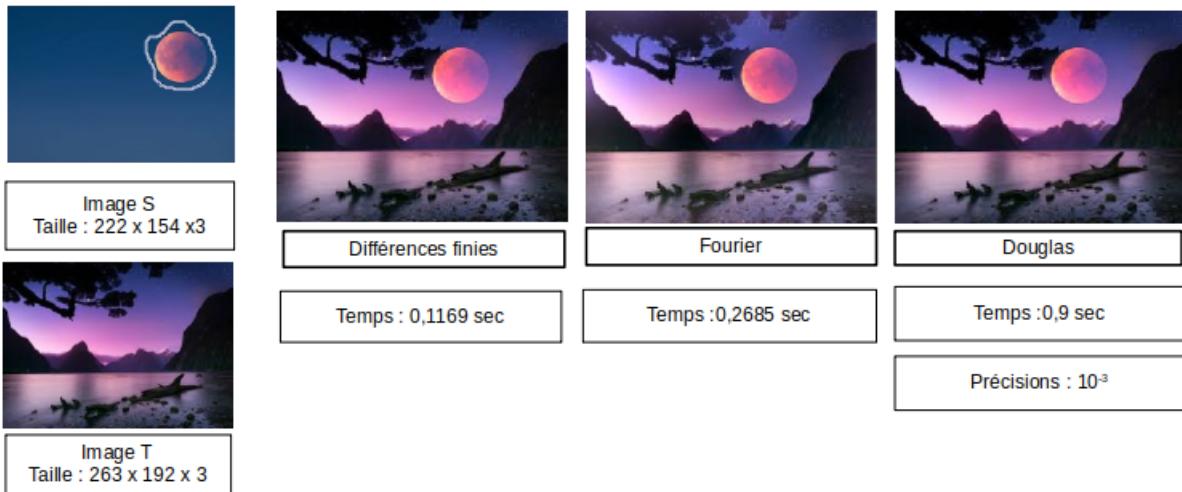


Figure 25: Résultats obtenus



Figure 26: Différence Finie ajustée

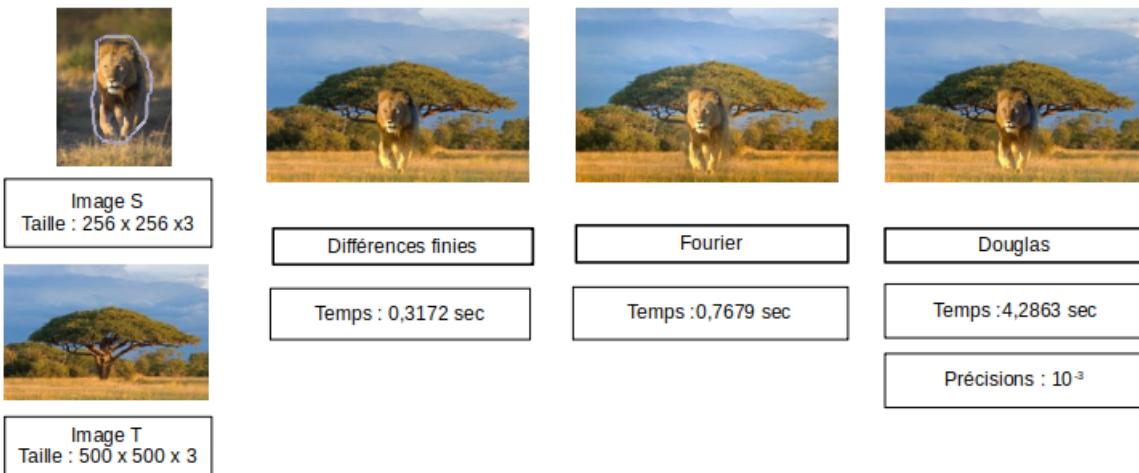


Figure 27: Résultats obtenus

6 Comparaison



Figure 28: Résultats obtenus

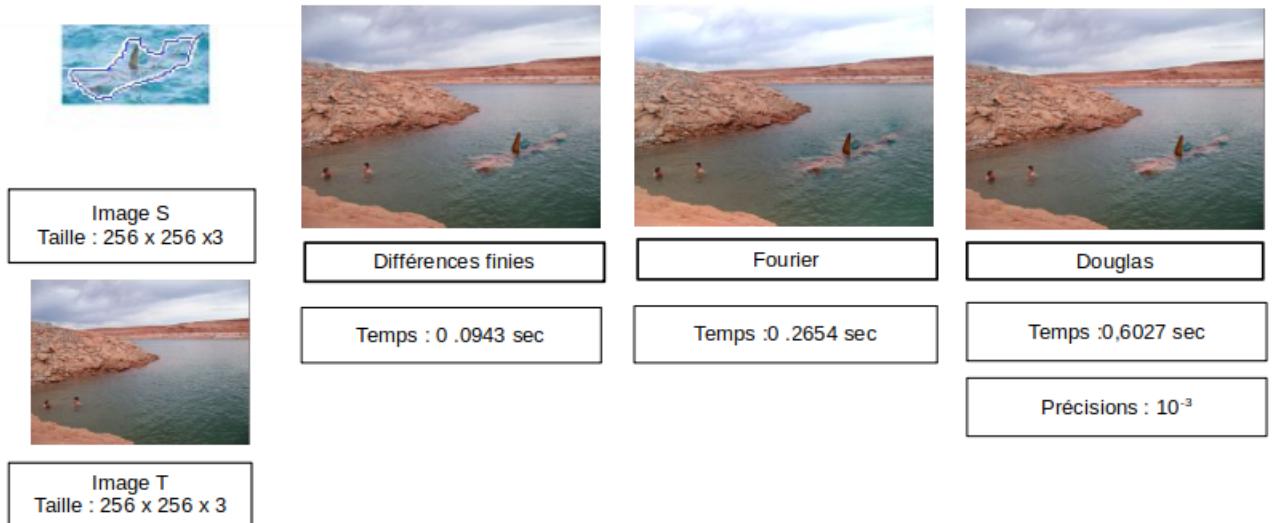


Figure 29: Résultats obtenus

6.1 Différences de résultat

En comparant les images ci-dessus, nous pouvons observer quelques différences entre la méthode de Fourier et celle des Différences finies. En effet, la méthode des différences finies ne travaille que sur une partie de l'image, celle qui va être collée, afin de l'adapter au mieux à l'image de fond. Le reste de l'image n'est pas modifié et on retrouve exactement l'image initiale T en dehors du domaine.

La méthode de Fourier elle, modifie toute l'image, elle n'adapte pas seulement la partie à coller, mais c'est toute l'image qui est modifiée. Fourier effectue un "mélange" des deux images.

Plus la précision de l'algorithme de Douglas augmente, plus le résultat obtenu avec celui-ci est proche de celui obtenu avec les différences finies.

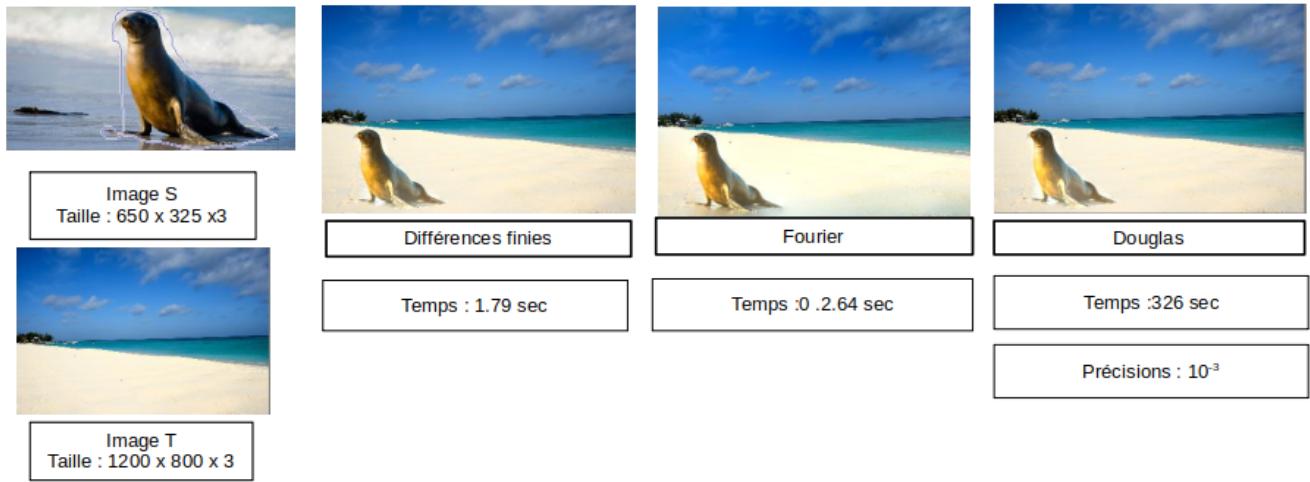


Figure 30: Résultats obtenus

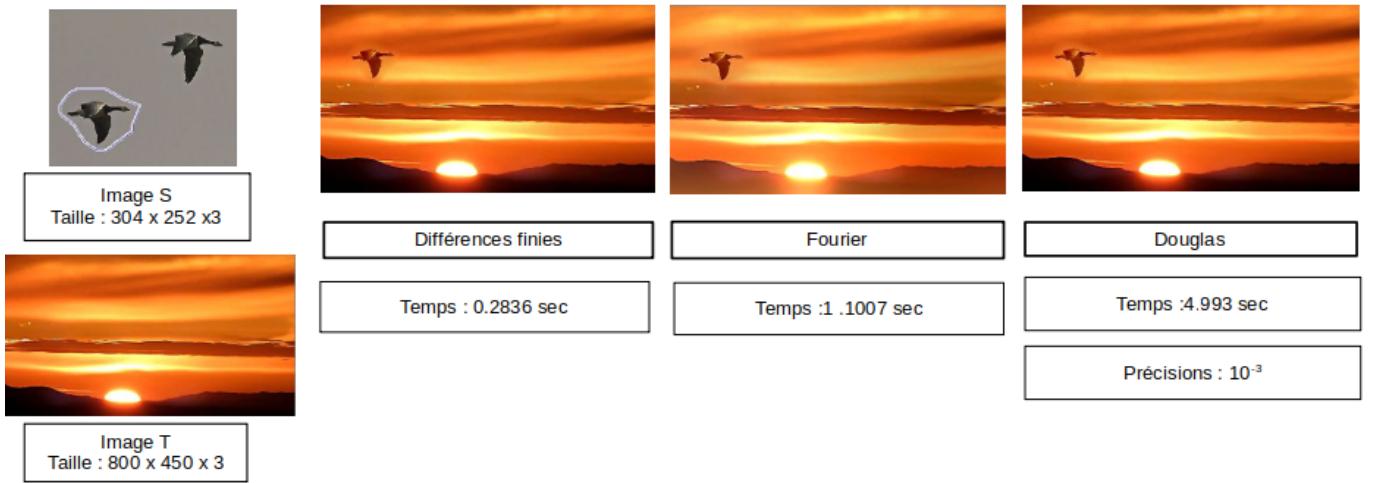


Figure 31: Résultats obtenus

6.2 Différence de temps

La méthode des différences finies fait intervenir une inversion matricielle, qui si elle est grande, augmentera significativement le temps de calcul de l'algorithme. Cette méthode consiste en la résolution d'un système plus ou moins grand, qui peut parfois prendre du temps.

Sur de grandes sélections, la méthode de Fourier, semble plus rapide, il est facile de calculer le gradient de l'image, et la fft ("fast fourier transform") est plus rapide. Sur de grandes sélections c'est donc cette méthode qui serait à privilégier.

La méthode de Douglas, est beaucoup plus lente, à cause du coût de l'opérateur proximal de f , et du nombre d'itérations répétées.

7 Implémentation

Voici une présentation de l'interface et ses fonctionnalités.

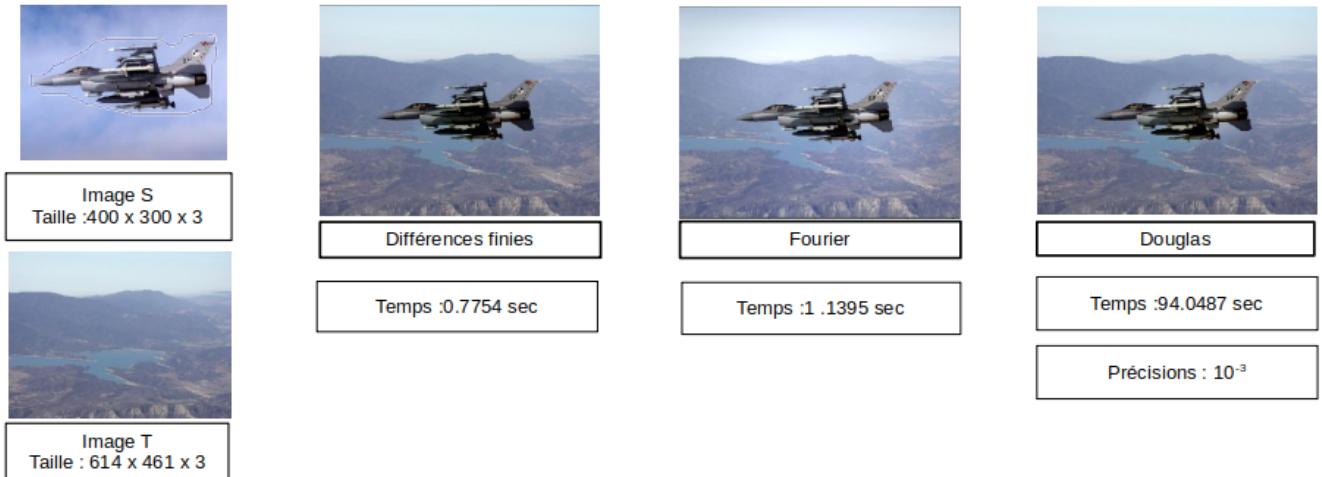


Figure 32: Résultats obtenus

7.1 Interface

Voici comment se présente notre interface : Elle est découpée en "blocs". Le premier bloc appelé "Open Button",



Figure 33: Interface 1.0

se situe en haut à gauche, il permet l'ouverture d'images. En cliquant sur "Open Image S", une boîte de dialogue s'ouvre et permet de charger une image, présente dans l'ordinateur. De même en cliquant sur le bouton T. Une fois ces images chargées elles s'affichent dans le bloc numéro 2 appelé "Images selected". L'image la plus haute dans ce bloc correspond à l'image S, tandis que l'image si situant en bas affiche l'image T. Il y a aussi un bouton "Save" permettant d'enregistrer l'image obtenue.

Une fois les images affichées, il faut sélectionner la région dite "à coller" dans l'image S, et la région où va se faire le collage dans l'image T.

Choix des méthodes Dans cette interface il est possible de sélectionner la méthode à utiliser. Le choix se porte sur 3 algorithmes :

- Les différences finies

- Fourier
- Douglas-Rachford

Il suffit de sélectionner la méthode pour l'appliquer. En cliquant sur le bouton "Paste !", présent dans le bloc "Open button", la nouvelle image est calculée et s'affiche dans le bloc "Main Result" situé en haut à droite.

Les options D'autres options s'offrent à vous dans cette interface. Il est possible d'effectuer l'amélioration vue plus haut, en la sélectionnant, case : "Change selection". Il est aussi possible de zoomer sur les différentes images, à l'aide de la case zoom. Enfin il est possible de travailler sur des images en couleur, en cochant la case adéquate. Toutes ces options se trouvent dans le bloc : Customizer.

Sliders Des sliders sont situés de parts et d'autres de l'image finale, dans le bloc "main result". En activant ces sliders, il est donc possible de bouger la sélection à l'intérieur de l'image. Ainsi, l'image finale est recalculée, et l'image à coller, sera collée à la position indiquée par les sliders.

bloc Results Le dernier bloc est un bloc d'affichage, il affiche une fois créée l'image obtenue par l'algorithme sélectionné, dans le bloc correspondant, cela permet une comparaison rapide.

7.2 Organisation du code

Nous avons organisé le code sous forme de classes. Le projet contient 4 classes principales, la classe Fourier, la classe DFinies, la classe Douglas et enfin la classe Mask. Notre projet contient aussi deux fonctions, la fonction copier/coller et la fonction du gradient conjugué.

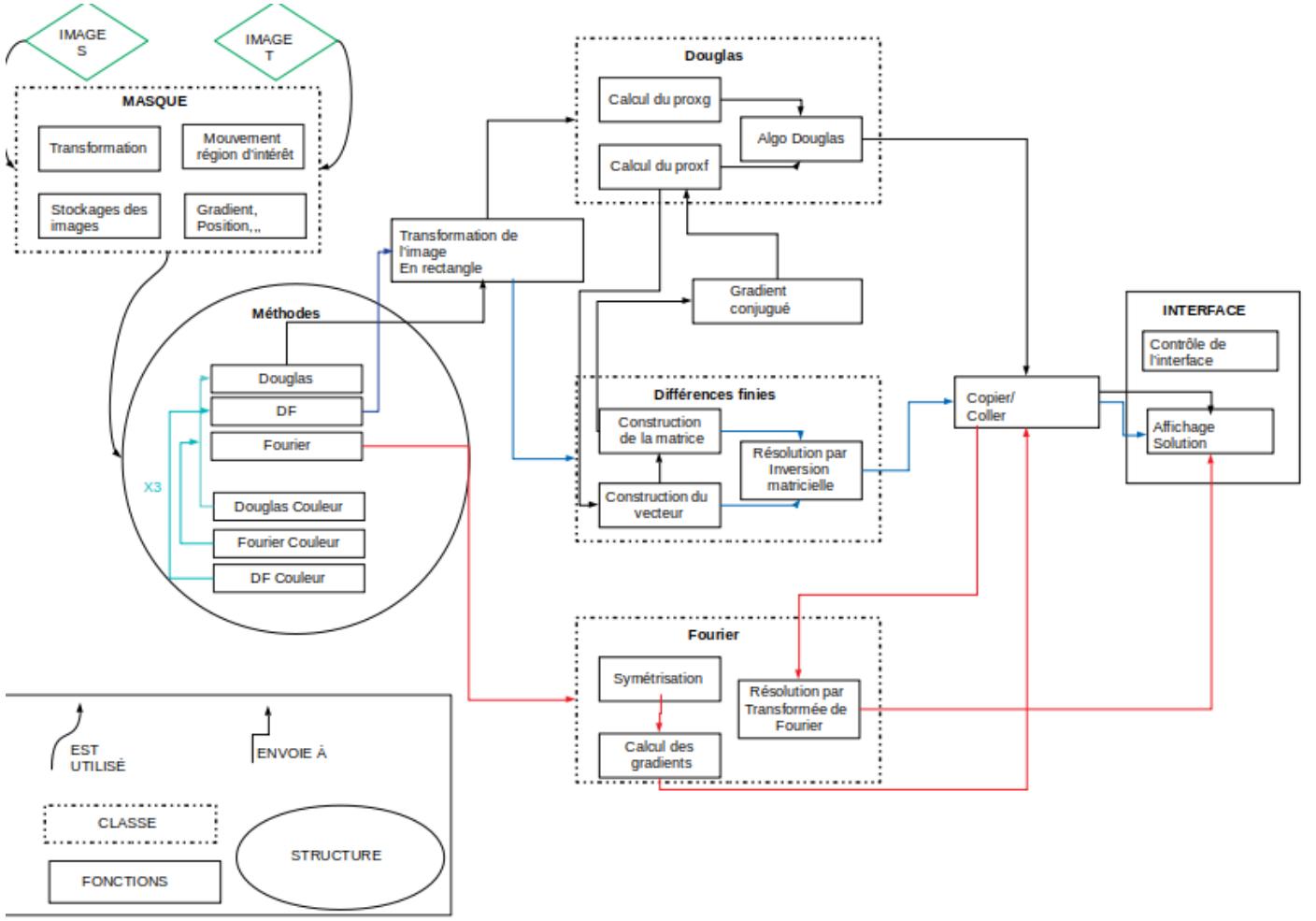


Figure 34: Structure et interactions du code

Explication du schéma Le code est organisé en différentes classes (4).

Douglas Cet algorithme étant plutôt long (en terme de temps), nous ne travaillerons pas sur l'image entière mais sur le plus petit rectangle autour de la sélection. En d'autre terme nous collerons l'image S sur une sous-image de T, que nous réinsérerons par la suite au bon endroit dans T. Afin de résoudre le problème à l'aide de l'algorithme de Douglas, nous avons crée une classe du même nom. A l'intérieur de celle-ci nous calculons les opérateurs proximaux nécessaires. Avec une particularité, l'opérateur proximal de la norme, nécessite la construction d'une matrice. Afin d'éviter les doublons nous utilisons donc la classe FDSystem, pour calculer cette matrice, et le vecteur associé. Puis nous inversons le système à l'aide de l'algorithme du gradient conjugué.

Différences finies Pour résoudre le problème avec les différences finies, la classe DFSystem, nous permet de calculer la matrice A, le vecteur b et enfin d'inverser le système pour trouver la solution. De la même manière que pour Douglas, nous ne travaillons pas sur l'image entière mais sur une sous-image que nous recollons au bon endroit à la fin.

Fourier Dans la classe Fourier nous symétrisons les images, puis calculons les gradients de celles-ci. Les images représentant les gradients sont par la suite fusionnées avant d'être envoyé à la fonction de résolution de la classe Fourier.

Mask La classe Mask, permet le traitement d'images et de masque, elle est mise à jour pendant tout le programme. C'est notamment elle qui permet le découpage d'image, la fusion de deux images, le redimensionnement des masques

si besoin.

La fonction gradient conjugué Elle permet de trouver la solution au système $Ax=b$ de manière plus rapide qu'une inversion dans matlab.

La fonction copier coller Elle écrase certains pixels d'une image par les pixels d'une autre.

8 Conclusion

L'équation de Poisson avec conditions aux bords de Dirichlet permet donc l'insertion d'une image dans une autre. A l'inverse, nous pouvons utiliser cette équations pour effacer certaines parties d'une image, ou encore pour augmenter la luminosité d'une image. Cette méthode est plutôt efficace même si elle possède ses limites. En effet, la couleur de l'objet à coller est par conséquent bien modifiée, qui parfois rends le collage beaucoup moins naturel. De même une grande sélection, est visible sur un fond très détaillé. Plus l'image de fond 'T' possède des variations, contours, détails, plus le collage devient mauvais.

Nous avons précédemment vu qu'une sélection plutôt large pouvait masquer des objets importants de l'image T. Ce problème peut être en partie résolu par des systèmes de comparaisons de gradients ou de moyenne mais le résultat n'est pas pas fiable, surtout si l'image à coller possède peu de variations. Cependant nous pouvons essayer d'étendre la résolution de cette équation sur des vidéos, en voulant incruster une vidéo dans une autre. Par exemple, réussir à incruster un vol d'oiseaux dans une vidéo. Si le vol se déplace dans la vidéo il faudrait être capable de détecter les mouvements de l'oiseau, (à l'aide de carte de disparité) afin de déplacer la sélection en même temps que l'objet.

References

- [1] Douglas-rachford method and admm. <http://www.seas.ucla.edu/~vandenbe/236C/lectures/dr.pdf>.
- [2] Méthode du gradient conjugué. https://fr.wikipedia.org/wiki/M%C3%A9thode_du_gradient_conjug%C3%A9.
- [3] Introduction to optimization. https://www.math.u-bordeaux.fr/~jaujol/PAPERS/optim_masterbdx.pdf, 2019.
- [4] J. Matias Di Martino, Gabriele Facciolo, and Enric Meinhardt-Llopis. Poisson image editing. https://www.ipol.im/pub/art/2016/163/article_lr.pdf, 2016.
- [5] Edoardo Provenzi. Transformée de fourier et ses applications. https://www.math.u-bordeaux.fr/~eprovenzi/include/Poly_TransFourier.pdf.