

EQUATION DE POISSON ET TRAITEMENT D'IMAGE

GROUPE DE TRAVAIL THÉMATIQUE

4TQMS801S

---

## **Equation de Poisson et traitement d'image**

---

Virginie MONTALIBET

Sébastien Eyzat

April 25, 2020



# Contents

<b>1 Présentation du problème</b>	<b>2</b>
1.1 Problème mathématique associé . . . . .	3
1.2 Résolution et équivalence avec l'équation de poisson . . . . .	4
<b>2 Résolution</b>	<b>5</b>
2.0.1 Qu'est qu'une image . . . . .	5
2.1 Discrétisation . . . . .	6
2.2 Notations . . . . .	6
2.3 Méthode des différences finies . . . . .	6
2.3.1 Discrétisation du Laplacien . . . . .	7
2.3.2 Résolution de l'équation de Poisson . . . . .	7
2.4 Exemple . . . . .	8
2.5 Fourier . . . . .	10
2.5.1 Rappel et définitions des opérateurs . . . . .	10
2.5.2 Résolution de la méthode Fourier . . . . .	11
<b>3 Implémentation</b>	<b>11</b>
3.1 Interface . . . . .	11
3.1.1 Ouverture d'images . . . . .	12
3.1.2 Sélection . . . . .	12
3.1.3 Choix des méthodes . . . . .	12
3.1.4 Différences finies . . . . .	12
3.1.5 Options . . . . .	12
3.1.6 Sliders . . . . .	13
3.2 Organisation du code . . . . .	13
<b>4 Résultats obtenus</b>	<b>14</b>
<b>5 Comparaison</b>	<b>14</b>
5.1 Différences de résultat . . . . .	15
5.2 Différence de temps . . . . .	16
<b>6 Optimisation</b>	<b>17</b>
6.1 Méthode de Douglas . . . . .	17
6.1.1 D'un problème avec contraintes... . . . . .	17
6.1.2 ... À un problème sans contraintes . . . . .	18
6.1.3 La convexité... . . . . .	18
6.1.4 ... Pour utiliser l'algorithme de Douglas... . . . . .	18
6.1.5 ... Avec les opérateurs proximaux ... . . . . .	18
6.1.6 Résultats obtenus . . . . .	19
6.1.7 Temps et coût de l'algorithme . . . . .	19
<b>7 Conclusion</b>	<b>19</b>

# 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 que nous devons résoudre, voici un exemple de ce que nous souhaitons faire.

Nous disposons des deux images présentées ci-dessous, l'image T(arget) et l'image S(ource).

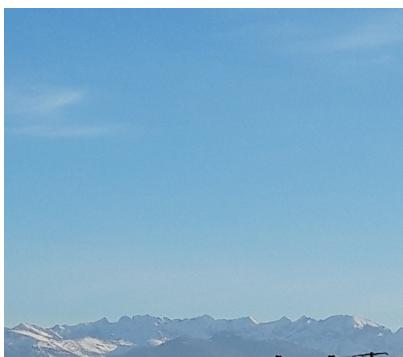


Figure 1: Image T



Figure 2: Image S

L'objectif est d'incruster toute ou une partie de l'image S dans l'image T. En terme de manipulations, cela consiste à effectuer un copier/coller ou encore un clônage de la seconde image dans la première. Le résultat que nous attendons pour une incrustation "réussie" est un résultat comme celui présenté ci-dessous :



Figure 3: Image finale attendue

Cette image est extraite d'une simulation effectuée sur '['demo.ipol.im/demo/163/'](#). Nous comparerons les images obtenues avec nos algorithmes avec celle-ci à la fin de ce rapport.

**Remarques** Ce résultat est réussi parce qu'il semble naturel, les frontières entre l'image collée et l'arrière plan sont très peu visibles et ont été estompées, les oiseaux présents dans la première image n'ont pas été déformé et semblent faire parti de l'image. Mais comment obtenir un tel résultat ?

Reprendons nos deux images séparées, T et S. Commençons par effectuer un simple copier/coller. Certains pixels de T sont donc écrasés par ceux de S. En effectuant cette manipulation voici l'image que nous devrions obtenir :



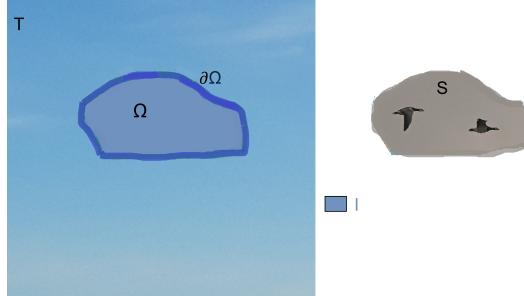
Figure 4: Simple copier/coller

Ce résultat n'est bien entendu pas convenable 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. Nous souhaitons obtenir un résultat 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 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, nous permet d'obtenir un bien meilleur résultat.

### 1.1 Problème mathématique associé

Comme écrit plus haut nous souhaitons 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 un "sous-domaine" correspondant à l'endroit du collage. Pour ce faire, considérons  $I$ , la partie de l'image finale à modifier. Nous pouvons ainsi représenter le problème sous forme schématique.



Ici :

- $T$  est l'image "Target", l'image destination, l'image sur laquelle s'effectuera le collage, l'arrière plan.
- $S$  est l'image "Source", l'image que nous souhaitons coller.
- $\Omega$  est le domaine dans lequel se trouvent nos inconnues.
- $\partial\Omega$  est la frontière de  $\Omega$ .
- $I$  est notre inconnue, la partie de l'image que nous ne connaissons pas et que nous voulons remplir.

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 notre image. Mais quelles sont les conditions qu'elle doit remplir pour que le rendu soit le meilleur possible? Notre nouvelle image  $I$  doit-elle être plus proche de l'image collée  $S$ , ou de l'image d'arrière-plan  $T$ ?

Pour répondre à cette question, reprenons le problème de départ.

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, nous devons garder les détails de l'image que nous voulons coller,  $S$ , ne pas modifier les variations qu'elle pourrait posséder comme par exemple, les contours ou les objets, lui appartenant. Nous devons donc pouvoir retrouver les informations présentes dans celle-ci. 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 le gradient de celle-ci. 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. Par conséquent, un fort changement d'intensité, implique un fort gradient. Au contraire, l'absence de variations entraîne un gradient presque nul. Le gradient d'une image permet entre autre de détecter les contours de celle-ci.

Nous souhaitons ici, que les contours de l'image finale soient très proches de l'image à coller. Mathématiquement, nous voulons donc une fonction  $I$  dont le gradient est le plus proche possible, ou identique à celui de l'image collée,  $S$ . 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  $T$ . Nous voulons donc, mathématiquement, chercher la fonction  $I$  qui vérifie:

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

Répondre au problème implique donc de résoudre un problème variationnel classique auquel des conditions sur le bord de Dirichlet sont ajoutées. Réécrivons le problème mathématique que nous cherchons 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 cette équation, nous devons trouver le minimum de la fonction  $g$  suivante :

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

Si  $g$  admet un minimum alors, celui-ci annule son gradient.

**Calcul de  $\nabla g(I)$**  À l'aide des formules de Taylor-Young à l'ordre 1 :

Soit  $f$  une fonction:  $\mathbb{R}^n \rightarrow \mathbb{R}$  et  $u \in \mathbb{R}^n$ , tel que  $\|u\| = 1$  :

$$f(x_0 + \epsilon u) = f(x_0) + \epsilon \langle \nabla f(x_0), u \rangle + o(\epsilon)$$

En appliquant ces formules à notre cas :

$$g(I + \epsilon u) - g(I) = \int_{\Omega} \|\nabla(I + \epsilon u) - v\|^2 - \|\nabla I - v\|^2 dx$$

En développant :

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

Par identification, le gradient de  $g$  vaut

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

Le minimum de  $g$  annule son gradient donc on cherche  $I$  tel que :

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

Ici  $v = \nabla S$

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

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

$$\Delta I = \Delta S$$

En remplaçant dans (1). On obtient le problème suivant :

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

qui n'est autre que l'équation de Poisson, avec conditions au bord de Dirichlet. Ainsi : résoudre le problème variationnel est équivalent à résoudre l'équation de poisson avec conditions aux bords de Dirichlet. Nous décrirons dans ce rapport 3 manières de résoudre cette équation.

## 2 Résolution

### 2.0.1 Qu'est qu'une image

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.

Avant de résoudre numériquement le problème, nous rappelons ce qu'est un image et le parcours d'une image. 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. Nous verrons donc une image comme la succession de pixels et nous pouvons la représenter comme une grille, dans laquelle chaque carré représente un pixel. Schématiquement, nous pourrions donc découper notre image comme ci-dessous. Nous



Figure 5: Maillage d'une image

numéroterons les pixels suivant la règle suivante. Le premier pixel est situé en haut en gauche, puis il suffit de parcourir la grille comme ci-dessous :

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

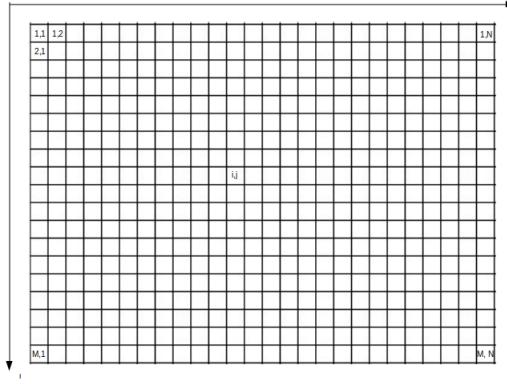


Figure 6: Parcours d'une grille

Rappelons que nous souhaitons résoudre l'équation de Poisson suivante :

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

## 2.1 Discrétisation

Commençons par résoudre le problème à l'aide de la méthode des différences finies. Afin de résoudre le problème (2), il faut commencer par discréteriser le laplacien de notre image.

## 2.2 Notations

Nous utiliserons certains opérateurs que nous définissons ici afin de ne pas alourdir les différentes sections.

**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 :  $\Delta$ , et : $\Delta = \operatorname{div}(\nabla f)$ .

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

## 2.3 Méthode des différences finies

Nous cherchons à résoudre :

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

En utilisant la méthode des différences finies, commençons par discréteriser le Laplacien. Pour discréteriser celui-ci, il faut donc commencer par discréteriser  $\frac{\partial^2 I}{\partial x^2}$  et  $\frac{\partial^2 I}{\partial y^2}$ .

**Discrétisation des dérivées secondes :** A l'aide des formules de Taylor à l'ordre 2 ci-dessous :

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

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

Et en effectuant la somme de ces deux équations, nous obtenons une discrétisation possible de  $\frac{\partial^2 I(x, y)}{\partial x^2}$ :

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

### 2.3.1 Discrétisation du Laplacien

Comme le Laplacien n'est autre que la somme de  $\frac{\partial^2 I(x, y)}{\partial x^2}$  et  $\frac{\partial^2 I(x, y)}{\partial y^2}$ , une discrétisation possible de celui-ci s'écrit de la forme :

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

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 de calculer 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	

### 2.3.2 Résolution de l'équation de Poisson

Soit  $g(x, y) = S(x+1, y) + S(x-1, y) + S(x, y+1) + S(x, y-1) - 4 \times S(x, y)$   
Résoudre  $\Delta I(x, y) = \Delta S(x, y)$  sur  $\Omega$  est équivalent à résoudre :

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

Nous devons donc résoudre un système à  $M \times N$  inconnues.

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

Afin de résoudre ce système, il est plus facile de l'écrire sous forme matricielle. Nous devons donc résoudre un système de la forme  $AI = b$  et sa solution est (en supposant que A est inversible) :  $I = A^{-1} \times b$ . 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 :

$$\begin{pmatrix} -4 & 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 \\ 1 & -4 & 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 \\ 0 & 1 & -4 & 1 & 0 & \dots & 0 & 1 & 0 & \dots \\ \dots & & & & & & & & & \\ 1 & 0 & \dots & 1 & -4 & 1 & 0 & \dots & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} I(1, 1) \\ I(2, 1) \\ \dots \\ I(1, 2) \\ I(2, 2) \\ \dots \\ I(M, N) \end{pmatrix} = \begin{pmatrix} g(1, 1) \\ g(2, 1) \\ \dots \\ g(1, 2) \\ g(2, 2) \\ \dots \\ g(M, N) \end{pmatrix} \quad (4)$$

**Transformation de l'image** Afin de pouvoir résoudre ce système, I doit être un vecteur colonne. Or dans notre cas, I est une matrice de taille  $(M, N)$ . Le nouveau vecteur I, est la concaténation des colonnes de l'image I, comme ci dessous.

**Ecriture de la matrice du système** La matrice A du système  $AI = b$ , est une matrice creuse, inversible. Nous verrons comment la remplir dans l'exemple ci-dessous.

**Ecriture de vecteur** Après avoir calculé le Laplacien de l'image S, le vecteur b est obtenu en concaténant les colonnes de l'image  $\Delta S$ .

## 2.4 Exemple

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

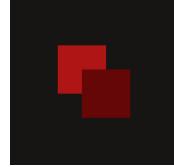


Figure 7: Image à coller

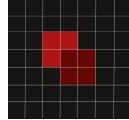


Figure 8: Vue grille pixel

image que nous nommerons T. Ici, notons  $\Omega$ , l'intérieur de la zone encadrée par la ligne verte, et des  $\partial\Omega$ , les pixels

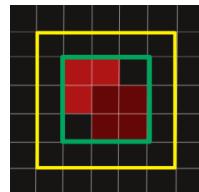


Figure 9: Sélection à coller

appartenant à  $J/\Omega$

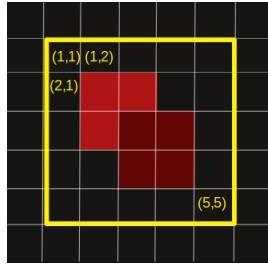


Figure 10: Numérotation des pixels

## Construction du système

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

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

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

Figure 11: Matrice du système

Le vecteur  $I$  est obtenu en concaténant les colonnes de la sélection, il est de taille(25 × 1) :

$$\begin{pmatrix} I(1, 1) \\ I(2, 1) \\ \dots \\ I(5, 1) \\ \dots \\ I(1, 3) \\ \dots \\ I(5, 3) \\ \dots \\ I(5, 5) \end{pmatrix} \quad (6)$$

Enfin le vecteur b

$$\begin{pmatrix} T(1,1) \\ \dots \\ \Delta S(2,2) \\ \dots \\ T(5,2) \\ \Delta S(1,3) \\ \dots \\ T(5,3) \\ \Delta S(2,4) \\ \dots \\ \Delta S(4,4) \\ T(5,4) \\ \dots \\ T(5,5) \end{pmatrix} \quad (7)$$

La solution I, s'écrit sous la forme  $I = A^{-1}b$ .

## 2.5 Fourier

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

### 2.5.1 Rappel et définitions des opérateurs

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

$$\widehat{S}(x, y) = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} S(k, l) e^{-2\pi i \left(\frac{k \times x}{M} + \frac{l \times y}{N}\right)} \quad (8)$$

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

$$S(k, l) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \widehat{S}(x, y) e^{2\pi i \left(\frac{xk}{M} + \frac{yl}{N}\right)} \quad (9)$$

**Gradient** Nous pouvons calculer le gradient d'une fonction dans le domaine de Fourier. Nous considérons toujours S la fonction que nous souhaitons étudier.

$$\widehat{\nabla(S)} = \left( \begin{array}{c} \widehat{\frac{\partial S}{\partial k}} \\ \widehat{\frac{\partial S}{\partial l}} \end{array} \right) \quad (10)$$

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

$$\begin{aligned} \widehat{\frac{\partial S}{\partial k}} &= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} S(x, y) e^{2\pi i \left(\frac{k \times x}{M} + \frac{l \times y}{N}\right)} \left( \frac{-\pi ix}{M} \right) \\ &= \left( \frac{2\pi ix}{M} \right) \widehat{S}(k, l) \\ \widehat{\frac{\partial S}{\partial l}} &= \left( \frac{2\pi iy}{N} \right) \widehat{S}(k, l) \end{aligned} \quad (11)$$

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

On a donc :

$$\begin{aligned} \widehat{\frac{\partial S}{\partial k}} &= \left( \frac{2\pi i}{M} x \right) \widehat{S} \\ \widehat{\frac{\partial S}{\partial l}} &= \left( \frac{2\pi i}{N} y \right) \widehat{S} \end{aligned} \quad (12)$$

## Laplacien

$$\begin{aligned}\widehat{\frac{\partial^2 S}{\partial k^2}} &= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} S(x, y) e^{2\pi i (\frac{k \times x}{M} + \frac{l \times y}{N})} \left(\frac{2\pi i x}{M}\right)^2 \\ &= \left(\frac{2\pi i x}{M}\right)^2 \widehat{S(k, l)} \\ \widehat{\frac{\partial^2 S}{\partial k^2}} &= \left(\frac{2\pi i x}{M}\right)^2 \widehat{S(k, l)}\end{aligned}\tag{13}$$

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

$$\widehat{\Delta S} = \left(\frac{2\pi i x}{M}\right)^2 \widehat{S} + \left(\frac{2\pi i y}{N}\right)^2 \widehat{S}\tag{14}$$

### 2.5.2 Résolution de la méthode Fourier

La résolution dans le domaine de Fourier, nécessite quelques changements. Cette méthode fonctionnant mieux sur un domaine rectangulaire, nous devons donc modifier le domaine  $\Omega$ .

Rappelons que nous voulons résoudre le problème suivant :

$$\begin{cases} \Delta I(x, y) = \Delta S(x, y) \text{ si } (x, y) \in \Omega \\ I(x, y) = T(x, y) \text{ si } (x, y) \notin \Omega \end{cases}\tag{15}$$

Afin d'obtenir un domaine régulier considérons  $R$ , le domaine de l'image entière. Afin de pouvoir utiliser Fourier, il faut que nous puissions travailler symétriquement en même temps sur les deux images.

Nous voulons donc que  $\nabla I$  soit très proche de  $\nabla S$  dans  $\Omega$  mais aussi que  $\nabla I$  soit très proche de  $\nabla T$  en dehors de  $\Omega$ . En notant  $V$  :

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

Nous devons donc résoudre l'équation suivante :

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

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.

Ainsi en calculant le laplacien dans le domaine de Fourier, on obtient :  $\widehat{\Delta I} = \widehat{\operatorname{div}(V)}$

$$\begin{aligned}\left(\frac{2\pi i x}{M}\right)^2 \widehat{I} + \left(\frac{2\pi i y}{N}\right)^2 \widehat{I} &= \left(\frac{2\pi i x}{M}\right) \widehat{\partial_1 V} + \left(\frac{2\pi i y}{N}\right) \widehat{\partial_2 V} \\ \left(\left(\frac{2\pi i x}{M}\right)^2 + \left(\frac{2\pi i y}{N}\right)^2\right) \widehat{I} &= \left(\frac{2\pi i x}{M}\right) \widehat{\partial_1 V} + \left(\frac{2\pi i y}{N}\right) \widehat{V} \\ \widehat{I} &= \frac{\left(\frac{2\pi i x}{M}\right) \widehat{\partial_1 V} + \left(\frac{2\pi i y}{N}\right) \widehat{\partial_2 V}}{\left(\left(\frac{2\pi i x}{M}\right)^2 + \left(\frac{2\pi i y}{N}\right)^2\right)}\end{aligned}\tag{17}$$

Afin de retrouver  $I$ , il suffit d'appliquer la transformée inverse, à l'équation ci-dessus.

## 3 Implémentation

Voici une présentation de l'interface et ses fonctionnalités.

### 3.1 Interface

Voici comment se présente notre interface :



Figure 12: Interface 1.0

### 3.1.1 Ouverture d'images

Vous pouvez ouvrir n'importe quelle image en cliquant sur les boutons situés en haut à gauche de l'interface. En cliquant sur eux, une boîte de dialogue vous proposera de choisir une image présente sur votre ordinateur. Une fois sélectionnée cette image sera affichée sur l'interface et vous pourrez travailler dessus. Image S correspond à l'image Source, c'est l'image que vous souhaitez coller. Et l'image T représente l'image Target, l'image d'arrière plan.

### 3.1.2 Sélection

Une fois vos deux images ouvertes, il vous faut sélectionner la partie de l'image S (8) que vous souhaitez coller dans l'image T. Pour cela cliquez une première fois sur l'image S : celle du haut puis cliquez une seconde fois pour dessiner sur l'image, la partie que vous souhaitez extraire.

Pour la seconde image, faites de même (9), cliquez une première fois sur l'image puis cliquez une seconde fois pour choisir l'endroit où vous souhaitez coller la partie sélectionnée plus tôt. Voici ce que vous devriez obtenir.

### 3.1.3 Choix des méthodes

Vous pouvez sélectionner la méthode avec laquelle vous voulez coller l'image S dans l'image T. Vous avez pour l'instant deux méthodes :

- Avec les différences finies (4)
- Avec Fourier (5)

### 3.1.4 Différences finies

Une fois la méthode DF choisie, il vous suffit de cliquer sur le bouton "Paste!"(3) pour afficher le résultat.

### 3.1.5 Options

Nous avons ajoutés des options comme l'option de zoom (6) qui permet de zoomer sur une image pour voir de plus près le collage de l'image.

**Amélioration avec change selection** Ainsi nous avons ajouté, l'option Change sélection qui re-dimensionne automatique la sélection afin de résoudre ce pb.

**Mode couleur** En sélectionnant le mode couleur, (8) vous pourrez travailler sur des images en couleur.

### 3.1.6 Sliders

Vous pouvez bouger les sliders présents à droite afin de déplacer la zone collée dans l'image de fond. La solution au problème est alors automatiquement recalculée en fonction de l'endroit où l'objet à collé.

## 3.2 Organisation du code

Nous avons organisé le code sous forme de classes. Le projet contient 2 classes principales, la classe, Fourier et la classe DFinies

### Classe Fourier

#### Propriétés

- $grad_Tj$ ,  $gradTi$  : permet de stocker le gradient de T, l'image d'arrière-plan
- $gradSj$ ,  $gradSi$  : Permet de stocker le gradient de S, l'image à coller.

#### Méthodes

- `compute_grad` : permet de calculer le gradient d'une image.
- `symmetry` : Permet d'étendre l'image par symétrie, alors la taille de l'image est multipliée par 2.  
`solve` : utilise la formule  $\|$ , pour trouver la solution de l'équation de Poisson à l'aide de la méthode de Fourier.
- `resize` : permet de redimensionner l'image qui avait été symétrisée auparavant.

### Classe DFinies

#### Propriétés

- `size` : taille de la matrice.
- `matrix` : permet de stocker la matrice du système.
- `vector` : permet de stocker le vecteur b associé au système.
- `i_vect` : Permet de stocker les numéros de lignes de la matrice creuse A
- `j_vect` : Permet de stocker les numéros des colonnes de la matrice creuse A
- `v_vect` : Permet de stocker les coefficients de la matrice creuse A

#### Méthodes

- `add_four` : ajoute à la matrice creuse le coefficient -4, à la position i, j
- `pix_up_is_inside` : ajoute le coefficient 1 à la matrice creuse, à la position
- `pix_down_is_inside` : ajoute le coefficient 1 à la matrice creuse, à la position
- `pix_left_is_inside` : ajoute le coefficient 1 à la matrice creuse, à la position
- `pix_right_is_inside` : ajoute le coefficient 1 à la matrice creuse, à la position
- `is_inside` :
- `find_roi` : Trouve la région d'intérêt dans l'image
- `find_useless` : Trouve les pixels, ne faisant pas partie de ma sélection
- `create_matrix` : Crée la matrice A, pour résoudre le système.

- *Laplacian* : Calcule le laplacien d'une image et met à jour le vecteur b.
- *solve* : Résout le système  $Ax=b$  et trouve l'image finale.

### Classe Mask

#### Propriétés

- *associate\_im*
- *associate\_roi*
- *matrix*
- *pos*
- *pos\_to\_move*
- *cut\_im*
- *shift\_done*
- *boundaries*

#### Méthodes

- *reinitialize\_mask*
- *save\_mask\_settings*
- *move\_roi*
- *invert\_mask*
- *find\_boundaries*
- *modify\_mask\_values*
- *adjust\_size*
- *change\_selection*

#### Fonction Copy/Paste

## 4 Résultats obtenus

## 5 Comparaison



Figure 13: Images sélectionnées



Figure 14: Différences finies



Figure 15: Fourier



Figure 16: Différence Finie ajustée



Figure 17: Images sélectionnées



Figure 18: Différences finies



Figure 19: Fourier

## 5.1 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 modifier 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 pour être la plus convenable possible.



Figure 20: Images sélectionnées



Figure 21: Différences finies



Figure 22: Fourier



Figure 23: Différence Finie zoom



Figure 24: Images sélectionnées



Figure 25: Différences finies



Figure 26: Fourier

## 5.2 Différence de temps

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

La méthode de Fourier, elle semble plus rapide, il est facile de calculer le gradient de l'image, et la fft est plus rapide. Sur de grandes sélections c'est donc cette méthode qui serait à privilégier.



Figure 27: Images sélectionnées



Figure 28: Différences finies



Figure 29: Fourier

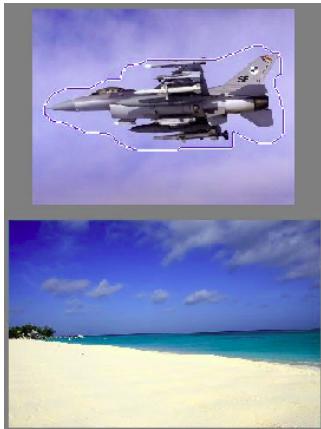


Figure 30: Images sélectionnées



Figure 31: Différences finies



Figure 32: Fourier

## 6 Optimisation

### 6.1 Méthode de Douglas

Nous avons vu deux manière de résoudre l'équation de poisson. Nous allons maintenant présenter une troisième méthode :

#### Le méthode de Douglas

Commençons par remarquer que l'équation que nous souhaitons résoudre est en réalité un problème d'optimisation avec contraintes.

Transformons ce problème, en un problème sans contraintes. Soit  $I$ , l'image à retrouver

#### 6.1.1 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.

### 6.1.2 ... 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 le problème :

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

avec

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

Nous avons bien équivalence entre notre problème sans contraintes et le problèmes (1). En effet, si  $I \in D \setminus \Omega$ , alors l'indicatrice vaut 0 et nous devons juste résoudre  $\min \int_{\Omega} \|\nabla I - \nabla S\|^2$ . Si au contraire  $I \notin D \setminus \Omega$ , alors nous devons minimiser quelque chose qui vaut plus  $+\infty$ . Le minimum n'existe pas. En effet, 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. Dans la suite nous prouverons que cette fonction est bien convexe, et par conséquent, qu'elle admet bien un minimum.

### 6.1.3 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  $\Omega$ , 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  $T$ , nous pouvons écrire les pixels de  $M$ , n'appartenant pas à  $\Omega$  de la manière suivante.

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  $\Omega$ , 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  $f+g$ , convexes, ainsi, la fonction  $F = f+g$  est elle aussi convexe. Elle admet donc un minimum. Nous pouvons ainsi résoudre numériquement ce problème.

### 6.1.4 ... 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 on a :

$$\begin{aligned} x_{k+1} &= \text{prox}_f(y_k) \\ y_{k+1} &= y_k + \text{prox}_g(2x_k + 1 - y_k) - x_{k+1} \end{aligned}$$

Nous devons donc calculer les opérateurs proximaux.

### 6.1.5 ... Avec les opérateurs proximaux ...

Un opérateur proximal est défini comme suit :

$$\text{prox}_f(x) = \arg \min_u \left\{ \frac{\|u - x\|^2}{2} + f(u) \right\}$$

## Opérateur proximal de $f$

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

Afin de faciliter les notations notons :

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

Nous cherchons donc

$$\operatorname{argmin}_u h(u)$$

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

En utilisant Taylor Young,

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

Par identification

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

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

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

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  $\operatorname{argmin}_u \{h(u)\}$  est équivalent à chercher  $\operatorname{argmin}_{u \in K} \left\{ \frac{\|u - x\|^2}{2} \right\}$ .

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

### 6.1.6 Résultats obtenus

#### 6.1.7 Temps et coût de l'algorithme

## 7 Conclusion