

EQUATION DE POISSON ET TRAITEMENT D'IMAGE

GROUPE DE TRAVAIL THÉMATIQUE

4TQMS801S

Equation de Poisson et traitement d'image

Virginie MONTALIBET

Sébastien Eyzat

April 27, 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	6
2.1.1 Notations	6
2.2 Méthode des différences finies	6
2.2.1 Résolution du système	7
2.3 Exemple	8
2.4 Fourier	10
2.4.1 Rappel et définitions des opérateurs	10
2.4.2 Résolution avec la méthode Fourier	11
3 Résultats obtenus	12
4 Comparaison	13
4.1 Différences de résultat	14
4.2 Différence de temps	15
5 Optimisation	15
5.1 Méthode de Douglas	15
5.1.1 D'un problème avec contraintes...	15
5.1.2 ... À un problème sans contraintes	15
5.1.3 La convexité...	16
5.1.4 ... Pour utiliser l'algorithme de Douglas...	16
5.1.5 ... Avec les opérateurs proximaux	16
5.1.6 Convergence de l'algorithme vers la solution	17
5.1.7 Résultats obtenus	18
5.1.8 Temps et coût de l'algorithme	19
5.2 Tableau comparatif	20
6 Implémentation	20
6.1 Interface	20
6.1.1 Ouverture d'images	20
6.1.2 Sélection	21
6.1.3 Choix des méthodes	21
6.1.4 Différences finies	21
6.1.5 Options	21
6.1.6 Sliders	21
6.2 Organisation du code	21
7 Conclusion	23

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).

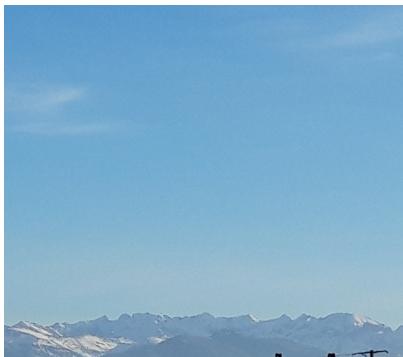


Figure 1: Image T



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, mais de manière . Le résultat attendu 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 à l'aide nos algorithmes avec celle-ci 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, les oiseaux présents dans la première image n'ont pas été déformés et semblent faire parti de l'image. 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 :



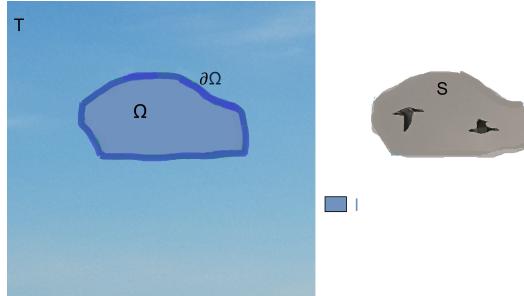
Figure 4: Simple copier/coller

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 clonage 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 trouvent 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.

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. Les oiseaux, l'avion ou la chaise à coller, doivent être présents dans l'image finale. 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 possiblement 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 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 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 :

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

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 : $v = \nabla S$.

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

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

Par identification, le gradient de g vaut

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

Le minimum de g annule son gradient, 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 :

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

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 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 celle-ci.

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 pixels. L'image peut donc être vues comme 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-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 5: Maillage d'une image

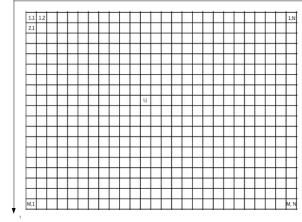


Figure 6: Parcours d'une grille

2.1 Discréétisation

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 :

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

Il est donc facile de voir que la somme de ces deux équations permet d'obtenir une discréétisation des 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 discréétisations 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 discréétisation 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 discréétisation du Laplacien :

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

	1	
1	-4	1
	1	

Figure 7: Laplacien du pixel

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 $I(i-1,j)$, $I(i+1,j)$, $I(i,j-1)$, $I(i,j+1)$.

Afin de trouver la solution au problème, il faut donc appliquer cette discréétisation à chaque pixel de l'image. Chaque pixel faisant intervenir ses voisins, la résolution du problème passe donc par la résolution d'une 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 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 :

$$\begin{cases} & I(1,1) = T(1,1) \\ & I(3,2) + I(1,2) + I(2,3) + I(2,1) - 4I(2,2) = g(2,2) \\ & I(3,3) + I(1,3) + I(2,4) + I(2,2) - 4I(2,3) = g(2,3) \\ & \dots \\ I(M,N-1) + I(M-2,N-1) + I(M-1,N) + I(M-1,N-2) - 4I(M-1,N-1) = g(M-1,N-1) \\ I(M,N) = T(M,N) \end{cases} \quad (3)$$

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, ainsi le blocs de la matrice correspondant aux pixels à l'intérieur du domaine sera rempli de la manière suivante :

$$\begin{pmatrix} -4 & 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & \dots & 0 \\ 1 & -4 & 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & \dots \\ 0 & 1 & -4 & 1 & 0 & \dots & 0 & 1 & 0 & \dots & \\ \dots & & & & & & & & & & \\ 1 & 0 & \dots & 1 & -4 & 1 & 0 & \dots & 0 & 1 & 0 \end{pmatrix} \quad (4)$$

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

$$\begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \dots \\ 0 & 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & \\ \dots & & & & & & & & & & \\ 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 \end{pmatrix} \quad (5)$$

Le vecteur b , contient les laplaciens des pixels de l'image S . Les valeurs des pixels de l'image S étant connus, il est facile de calculer le Laplacien de celle-ci en utilisant les discrétilisations vues ci-dessus. Le vecteur b , n'est autre que la concaténation des colonnes du Laplacien de S .

$$\begin{pmatrix} g(1,1) \\ g(2,1) \\ \dots \\ g(1,2) \\ g(2,2) \\ \dots \\ g(M,N) \end{pmatrix} \quad (6)$$

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

$$\begin{pmatrix} I(1,1) \\ I(2,1) \\ \dots \\ I(1,2) \\ I(2,2) \\ \dots \\ I(M,N) \end{pmatrix} \quad (7)$$

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 image que nous nommerons T .

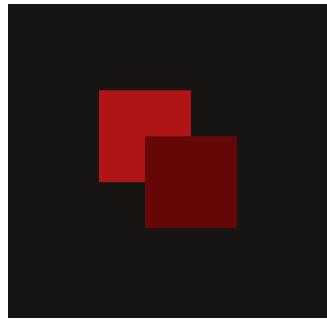


Figure 8: Images à coller

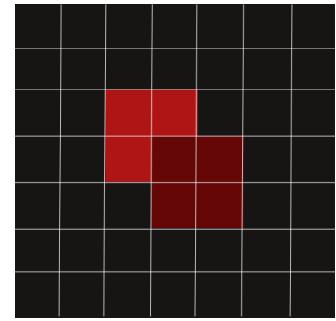


Figure 9: Vue grille pixel

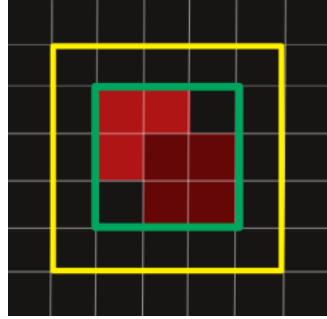


Figure 10: Sélection à coller

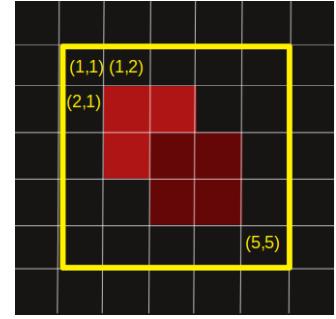


Figure 11: Numérotation des pixels

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

Construction du système

$$\left\{ \begin{array}{l} I_{1,1} = T_{1,1} \\ \dots \\ I_{5,1} = T_{5,1} \\ I_{1,2} = T_{1,2} \\ \Delta I_{2,2} = \Delta S_{2,2} \\ \Delta I_{3,2} = \Delta S_{3,2} \\ \Delta I_{4,2} = \Delta S_{4,2} \\ \dots \end{array} \right. \quad (8)$$

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

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

Figure 12: Matrice du système

$$I = \begin{pmatrix} I(1,1) \\ I(2,1) \\ \dots \\ I(5,1) \\ \dots \\ I(1,3) \\ \dots \\ I(5,3) \\ \dots \\ I(5,5) \end{pmatrix} \quad b = \begin{pmatrix} T(1,1) \\ \dots \\ \Delta S(2,2) \\ \dots \\ T(5,2) \\ \Delta S(1,3) \\ \dots \\ T(5,3) \\ \Delta S(2,4) \\ \dots \\ \Delta S(4,4) \\ T(5,4) \\ \dots \\ T(5,5) \end{pmatrix} \quad (9)$$

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

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

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. Nous allons maintenant voir une seconde méthode, plus rapide, en nous plaçant dans le domaine de Fourier.

2.4 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

2.4.1 Rappel et définitions des opérateurs

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

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

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

$$F(k, l) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \hat{F}(x, y) e^{2\pi i (\frac{xk}{M} + \frac{yl}{N})} \quad (11)$$

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 le Laplacien de la transformée de Fourier d'une fonction , et donc le gradient de celle-ci. F est toujours la fonction que nous souhaitons étudier.

$$\widehat{\nabla(F)} = \begin{pmatrix} \widehat{\frac{\partial F}{\partial k}} \\ \widehat{\frac{\partial F}{\partial l}} \end{pmatrix} \quad (12)$$

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

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

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

On a donc :

$$\begin{aligned} \widehat{\frac{\partial F}{\partial k}} &= \left(\frac{2\pi i}{M} x \right) \widehat{F} \\ \widehat{\frac{\partial S}{\partial l}} &= \left(\frac{2\pi i}{N} y \right) \widehat{F} \end{aligned} \quad (14)$$

Laplacien

$$\begin{aligned} \widehat{\frac{\partial^2 F}{\partial k^2}} &= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} F(x, y) e^{2\pi i (\frac{k \times x}{M} + \frac{l \times y}{N})} \left(\frac{2\pi i x}{M} \right)^2 \\ &= \left(\frac{2\pi i x}{M} \right)^2 \widehat{F(k, l)} \\ \widehat{\frac{\partial^2 F}{\partial k^2}} &= \left(\frac{2\pi i x}{M} \right)^2 \widehat{F(k, l)} \end{aligned} \quad (15)$$

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

$$\widehat{\Delta F} = \left(\frac{2\pi i x}{M} \right)^2 \widehat{F} + \left(\frac{2\pi i y}{N} \right)^2 \widehat{F} \quad (16)$$

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 :

$$\begin{cases} \Delta I(x, y) = \Delta S(x, y) \text{ si } (x, y) \in \Omega \\ I(x, y) = T(x, y) \text{ si } (x, y) \notin \Omega \end{cases} \quad (17)$$

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

Nous voulons donc que ∇I soit très proche de ∇S dans Ω mais aussi que ∇I soit très proche de ∇T en dehors de Ω . En notant V :

$$V = \begin{cases} \nabla S(x, y) & \text{si } (x, y) \in \Omega \\ \nabla T(x, y) & \text{si } (x, y) \notin \Omega \end{cases} = \begin{pmatrix} V_1 \\ V_2 \end{pmatrix} \quad (18)$$

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

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

$$\widehat{I} = \frac{\left(\frac{2\pi ix}{M}\right) \widehat{V}_2 + \left(\frac{2\pi iy}{N}\right) \widehat{V}_1}{\left(\left(\frac{2\pi ix}{M}\right)^2 + \left(\frac{2\pi iy}{N}\right)^2\right)} \quad (20)$$

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

3 Résultats obtenus



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



Figure 20: Images sélectionnées



Figure 21: Différences finies



Figure 22: Fourier

4 Comparaison



Figure 23: Différence Finie zoom



Figure 24: Images sélectionnées



Figure 25: Différences finies



Figure 26: Fourier



Figure 27: Images sélectionnées



Figure 28: Différences finies



Figure 29: Fourier

4.1 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.



Figure 30: Images sélectionnées



Figure 31: Différences finies



Figure 32: Fourier

4.2 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.

La méthode de Fourier, elle 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.

5 Optimisation

5.1 Méthode de Douglas

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 :

5.1.1 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.

5.1.2 ... 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

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

avec

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

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 .

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 :

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

5.1.3 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 écrire les pixels de M , n'appartenant pas à Ω 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.

5.1.4 ... 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 :

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

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

5.1.5 ... 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

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

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,

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

Par identification

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

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 .

5.1.6 Convergence de l'algorithme vers la solution

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

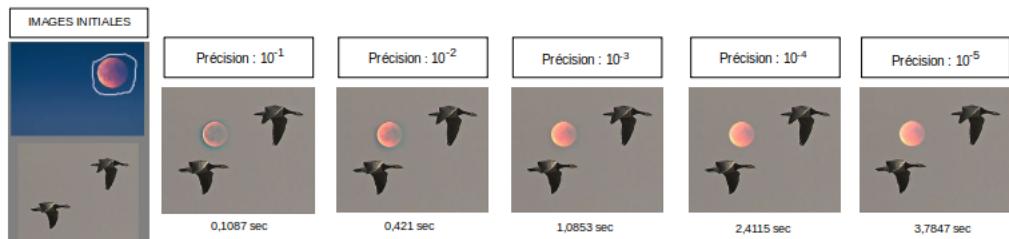


Figure 33: Convergence de l'algorithme vers la solution

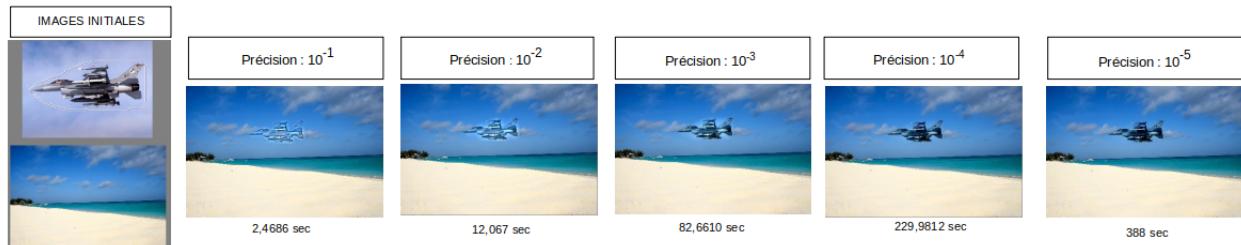


Figure 34: Convergence de l'algorithme vers la solution

5.1.7 Résultats obtenus

Avec une précision de 10^{-5}

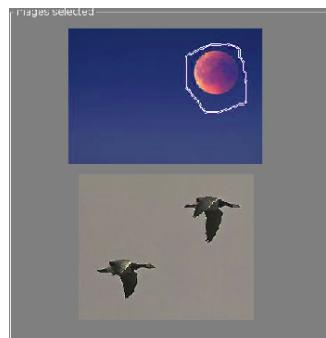


Figure 35: Images sélectionnées



Figure 36: Douglas, avec une précision de 10^{-5}

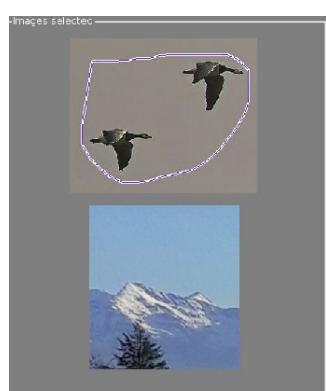


Figure 37: Douglas



Figure 38: Douglas, avec une précision de 10^{-5}

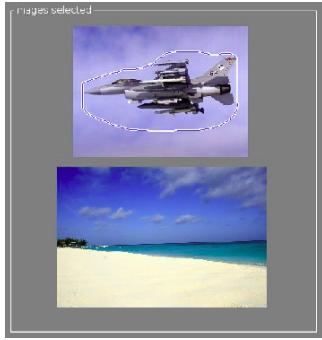


Figure 39: Images sélectionnées



Figure 40: Douglas, avec une précision de 10^{-5}

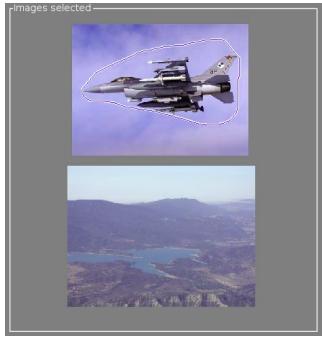


Figure 41: Images sélectionnées



Figure 42: Douglas, avec une précision de 10^{-5}

5.1.8 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, 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, car il semble converger plus lentement à partir de 10^{-3} , ce qui augmente considérablement le temps de calcul.

Afin de réduire ce temps de calcul il faudrait donc que l'algorithme converge 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.

5.2 Tableau comparatif

Tailles de l'image S	Taille de l'image T	Temps Douglas	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	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	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	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	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	0.0892 secondes	0.1754 secondes

Les résultats que nous obtenons avec les différentes images sont à peu près similaires. Nous pouvons bien sûr raccourcir le temps de calcul de l'algorithme de Douglas-Rachford, en diminuant la précision. Mais dans ce cas, les résultats obtenus avec celui-ci sont moins bons, et non comparables avec ceux des deux autres méthodes.

6 Implémentation

Voici une présentation de l'interface et ses fonctionnalités.

6.1 Interface

Voici comment se présente notre interface :

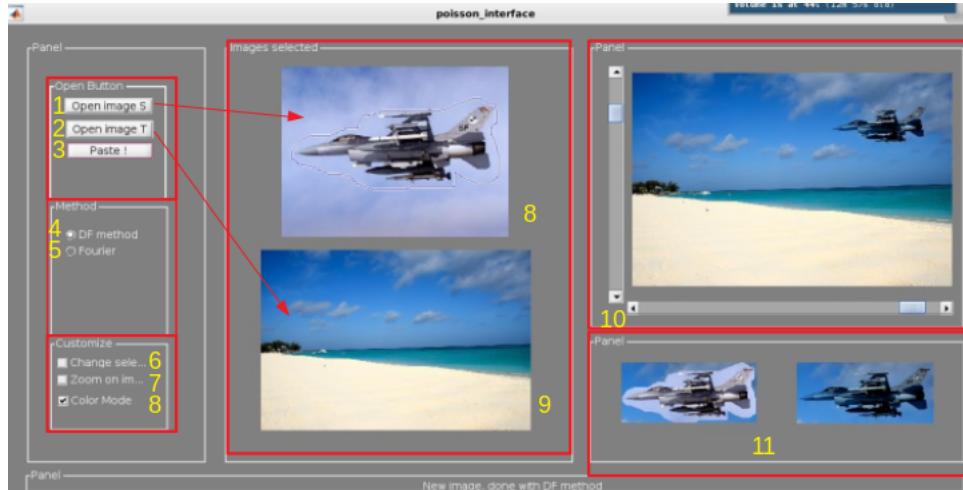


Figure 43: Interface 1.0

6.1.1 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.

6.1.2 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.

6.1.3 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)

6.1.4 Différences finies

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

6.1.5 Options

Nous avons ajouté 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.

6.1.6 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.

6.2 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 fonctions copier/coller et la fonction du gradient conjugué.

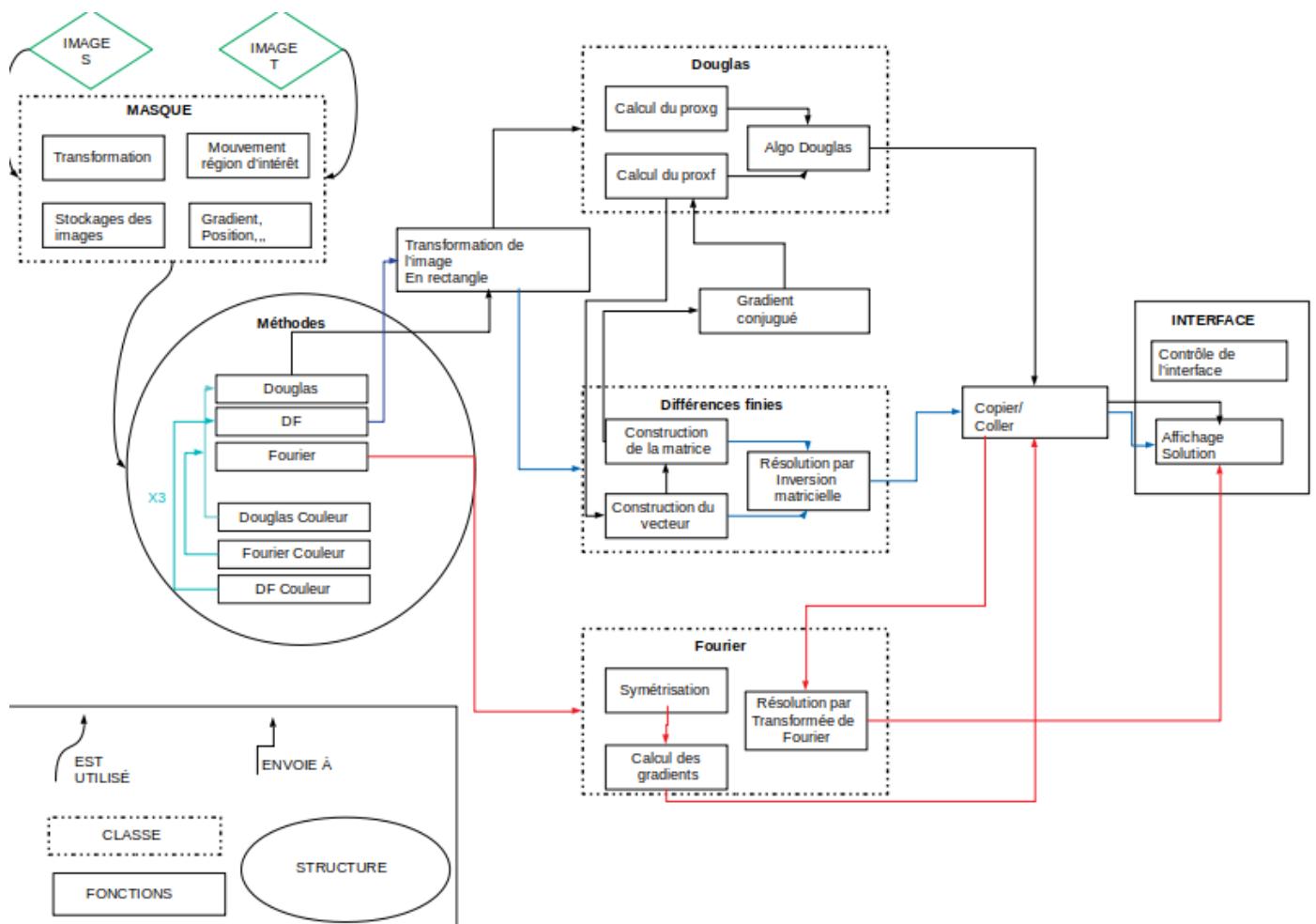


Figure 44: Structure et interactions du code

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.

7 Conclusion