TIPE
Traitement d'images satellitaires

Critères de qualité et méthode du Pan-Sharpening

Aymeric SANNIER
Nicolas VERHELST

2014-2015-2016

Table des matières

Chapitre 1

Introduction

Une partie non négligeable des satellites artificiels gravitant autour de la Terre ont pour objectif de réaliser des prises de vue de sa surface, tout d'abord dans le cadre d'études scientifiques puis plus récemment, dans un but de diffusion au grand public. L'institut Géographique National (IGN), met à disposition des professionnels et des particuliers les satellites Pléiades du CNES (Centre National d'Etudes Spatiales) afin de fournir de telles prises de vue. Ce satellite utilise une méthode particulière de traitement de l'image dite du *pan sharpening* afin de fusionner des images couleur basse définition avec des images monochrome haute définition.

Un exemple d'image traitées issues de Pléiades, disponible au grand Public sur *www.geoportail.gouv.fr* :

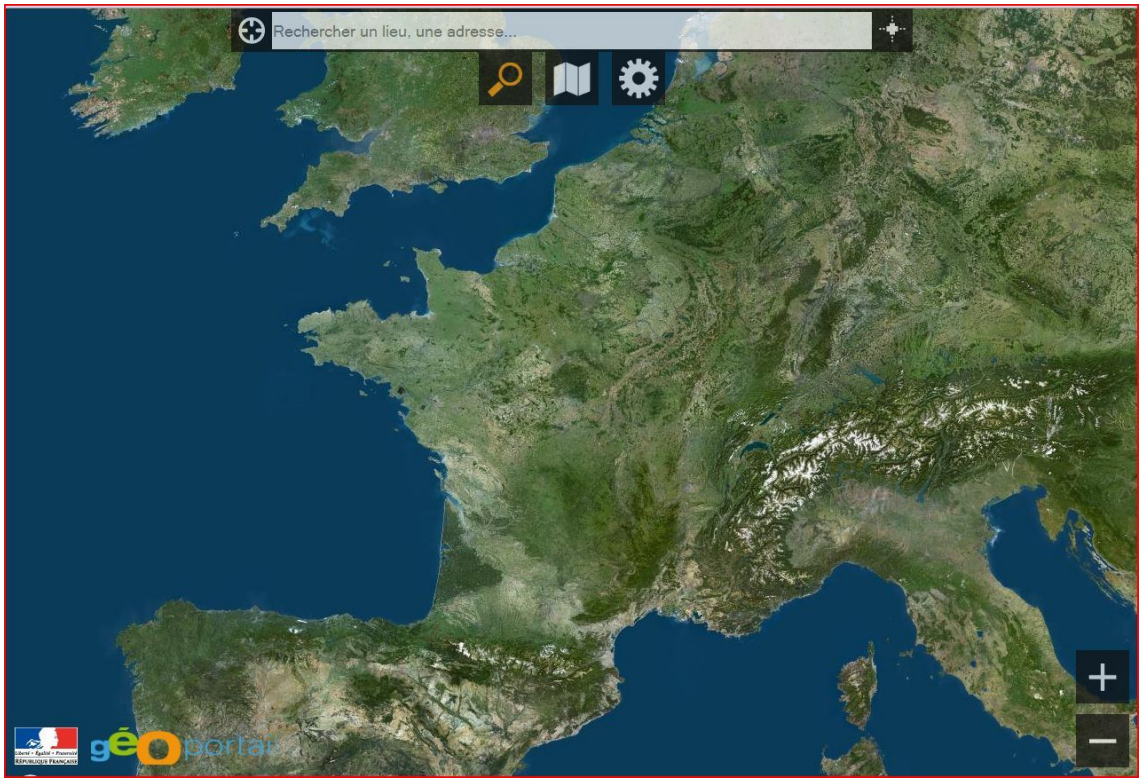

Figure 1.1 – Image satellite de la France disponible sur le site géoportail

Les deux satellites Pléiades qui nous fourniront les images traitées dans ce projet sont composés d'un système complexe de prise de vue que nous décriront plus loin. Pour notre TIPE, nous avons décidé de nous intéresser à la partie concernant la gestion de l'image une fois parvenue au sol sous forme de fichiers de bandes spectrales autrement dit, la chaîne de traitement au sol.

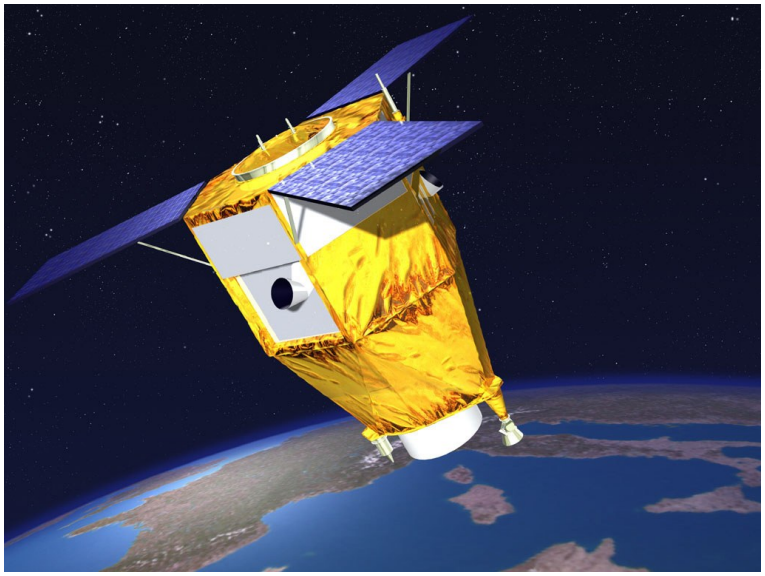

Figure 1.2 – Image de synthèse du satellite en vol

1.1 Les différents domaines d'application des satellite Pléiades

Les satellites pléiades sont au nombre de deux : Pléiade-1A et Pléiade-1B. Chacun de ces deux satellite de 980 kg a été lancé par Soyouz (depuis le centre spatial guyanais) pour orbiter à une une altitude de 694 km (orbite héliosynchrone [1]). Ils ont été créés dans un but de télédétection civile et militaire pour l'imagerie visible et le proche infrarouge (le temps d'utilisation des satellite est répartie selon ces deux instance). Ce télédétecteur *très haute résolution* (THR) est utilisé dans les domaines suivants :

- **Aménagement** : détection et identification d'éléments inférieurs à $1m^2$: véhicules mobiliers urbains, réseaux de voirie, buisson isolés.
- **Agriculture** : gestion des espaces et de la production agricole, repérage de zones de maladies des cultures.
- **Urbanisme et démographie** : localisation de constructions individuelles.
- **Défense** : recueil de renseignements dérivés des images et planification tactique.
- **Sécurité civile** : prévention, assistance durant les crises et évaluation post-crise notamment en cas de séisme.
- **Hydrologie** : topographie et études des pentes des bassins versants.
- **Forêts** : déforestation illégales et gestion de la production sylvicole.
- **Mer et littoral** : reconnaissance de navires et pollutions.
- **Génie civil** : tracés routiers, ferrés et oléoducs.

1.2 Système de prise de vue des satellites Pléiades

Les satellites pléiades on une très grande capacité de détection : Un pixel de l'image finale correspondra à 70cm pour la bande monochromatique (nuance de gris) et 2,8m pour le mode multispectrale (couleur). Cette précision est du aux capteurs numériques, mais aussi au télescope. Le télescope utilisé sur Pléiade est un télescope de type Cassegrain (composé de quatre miroirs) : c'est un télescope Korsch. La photo et la représentation schématique sont données ci-dessous :

1. Orbite géocentrique légèrement rétrograde dont on choisit l'altitude et l'inclinaison de façon que l'angle entre le plan d'orbite et la direction du soleil demeure quasiment constant. (Un satellite héliosynchrone repasse quotidiennement au-dessus d'un lieu à une heure solaire identique).

Figure 1.3 – Télescope Korsch dans un laboratoire de Thales.

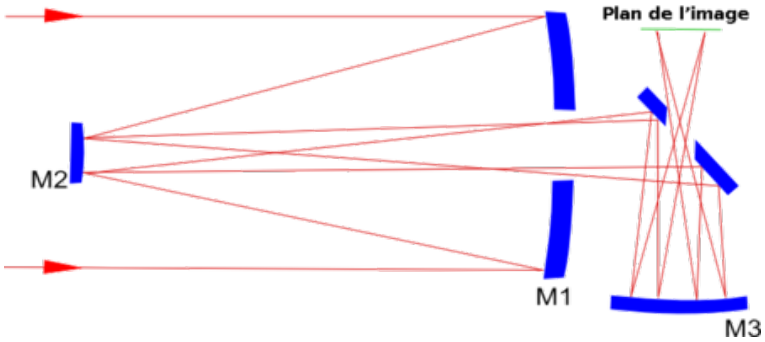

Figure 1.4 – Schéma du Télescope Korsch : type Cassegrain.

Les différentes bandes de détection du satellite son données ci-après :

Mode	Canal	Bande spectrale
Multispectrale	1	430 - 550 nm (bleu)
	2	490 - 610 nm (vert)
	3	600 - 720 nm (rouge)
	4	750 - 950 nm (proche infrarouge)
Panchromatique	P	480 - 830 nm (noir et blanc)

Table 1.1 – Différentes bandes de détection du satellite.

La largeur nominale de la prise de vue est de 20km. Mais qu'en est-il de la "longueur" de l'image ? C'est ici que se trouve la subtilité de la prise de vue : à la différence de l'appareil photo de monsieur tout le monde (qui comporte un capteur CCD [2] prenant l'intégralité de l'image dans un unique lapse de temps), les satellites Pléiade comportent un capteur qui ne reçoit la lumière que d'une unique ligne de la surface du globe. C'est alors le déplacement de cette ligne à la surface de la Terre (grâce à la rotation du satellite par rapport au sol) qui va permettre de créer une image : c'est un balayage.

On comprend alors, qu'en fonction de différents paramètres (trajectoire instable, mouvement dû au changements de température...), l'image va comporter quelques défauts : problèmes d'alignement,

2. Charge-Coupled Device, ou en français "dispositif à transfert de charge" (DTC)

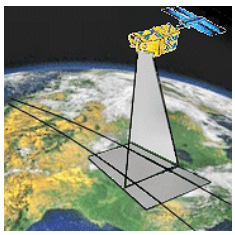

Figure 1.5 – Balayage de la surface du sol par le satellite

tremblement... De plus nous allons parfois trouver quelques pixels morts [3] (qui donnerons une ligne noir), etc...

1.3 Architecture numérique d'une image

Une image numérique est divisée en des millions pixels ; des sortes de carrés (ou bien parfois des rectangles) eux même subdivisés en trois [4] ou en quatre [5] canaux de données codés sur 8 ou 16 bits par des nombres entiers. Ces données prennent la forme de nombres entiers allant de 0 à 256 pour le 8 bits ou de 0 à 65536 pour le 16 bits. Ces trois canaux permettent de décrire l'image par trois couleurs : le rouge, le vert et le bleu (c'est le RVB) dont la combinaison en intensité permet l'affichage d'un nombre fini de couleurs [6]. Un quatrième canal à été mis en place dans le cas des formats gif et png afin de déterminer la transparence du pixel. Ces deux derniers formats ne seront ni décrits ni utilisés pour notre étude.

Ainsi, une image peut être ramenée à une matrice de dimension (profondeur) 3 et de la taille suivante :

(nombre de colonnes de l'image × nombre de lignes de l'image)

Chaque dimension correspondra à chacune des couleurs, alors que les coordonnées "plane" dans la matrice correspondront à la position du pixel dans l'image.

R	V	B	R	V	B
255	255	255	32	178	170
R	V	B	R	V	B
0	255	0	255	0	0

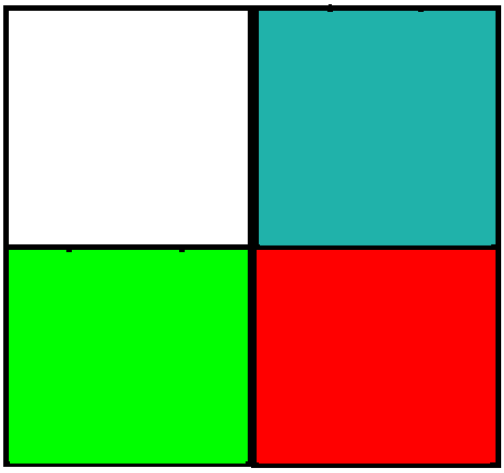

Figure 1.6 – Architecture d'une image numérique (rendu final à droite)

Pour les images en niveaux de gris, les pixels sont codés soit par un unique canal de luminance [7], soit

3. Pixels hors service.
4. Pour ce qui concerne les formats jpeg et tiff
5. Pour ce qui concerne les formats gif et png
6. On retiendra une possibilité de 16 million de couleur pour les images codées en 8 bits (codage de 0 à 255)
7. Plus la valeur de la luminance est élevée, plus le pixel est blanc et inversement

par les trois canaux couleur RVB. Dans ce cas, les trois canaux auront la même valeur : plus elle sera élevée, plus le pixel sera clair.

1.4 Problématique

Devant absolument établir une problématique claire et précise, ne ne pouvions pas traiter un sujet aussi vaste que toute la chaine de traitement d'images satellitaires. Nous avons cependant pensé à différents sujets : la mise en orbite du satellite, le maintient de la trajectoire du satellite sur son orbite, la protection extérieur du télescope face à la rudesse de l'espace, l'alimentation électrique du satellite, la communication avec le sol, le positionnement du satellite, etc... Certains de ces problèmes étaient trop difficiles à exploiter alors que d'autre ne laissaient pas suffisamment de place l'utilisation de nos connaissance apprises en cours – autant en mathématique qu'en physique-chimie ou qu'en science de l'ingénieur. Un centre d'intérêt commun nous a alors permis de faire notre choix : aimant tout deux la photographie il devenait évident de nous pencher sur l'image satellite en elle même. Tous les deux nous étions intrigués par la confection et l'amélioration de l'image. C'est alors avec cette optique que nous avons élaboré la problématique :

Problématique Dans le cadre d'une chaine de traitement d'image satellitaires, comment obtenir une méthode de *pan sharpening* efficiente et comment évaluer la qualité des images fusionnées en étudiant la dynamique de celle-ci, selon leur utilisation.

Par "*pan sharpening*" nous entendons la méthode qui permet de fusionner une image en nuance de gris haute définition avec une image couleur basse définition. Wikipédia nous donne la définition suivante :

> Pansharpening is a process of merging high-resolution panchromatic and lower resolution multispectral imagery to create a single high-resolution color image.

Par étude de la dynamique de l'image nous entendons la détermination de critère objectifs permettant de quantifier la qualité de l'image (par exemple : la définition, le contraste, la température, la saturation, le flou, la clarté, etc...).
Par utilisation de l'image nous pensons à deux utilisations particulières :

1. L'utilisation de l'image pour le **grand public**. Dans ce cas l'image devrai avoir des couleurs vives avec un contraste élevé, avec une grande saturation, de la clarté, et un bon contraste.
2. L'utilisation de l'image dans un **cadre professionnel**. Dans ce cas l'image devra représenter au mieux la réalité du terrain imagé, ceci autant sur le plan de la couleur que de la géométrie.

Chapitre 2

Objectifs

2.1 Objectif général

Nous avons fait le choix de scinder notre projet sur les deux années de scolarité en prépa. La première année nous nous somme chargés de créer des moyens d'évaluer la qualité d'images, alors que la deuxième année nous nous sommes penchés sur la création d'algorithmes de fusion menant au pan-sharpening de l'image à proprement parlé.

2.1.1 Objectif première année (MPSI)

Dans le but d'évaluer les différentes méthodes de pan-sharpening que nous mettrons en oeuvre par la suite, nous avons mis en place des algorithmes permettant d'évaluer les images selon les critères de qualité suivants :

- Vérité ;
- Contraste ;
- Netteté.

Divers procédés ont étés employés pour répondre à ces besoins. Nous les avons automatisés à l'aide du langage Python et de deux de ses principaux modules, Numpy et Scipy. Nous détaillerons l'utilisation de Python, de son IDE, ainsi que de ses module dans le chapitre des ressources utilisées.

2.1.2 Objectif deuxième année (MP)

C'est en deuxième année que nous avons abordé "l'âme du sujet" : le traitement de l'image par différents algorithmes de *pan sharpening*. Le travail envisagé pour répondre à notre problématique était alors de trouver et de programmer différents algorithmes de *pan sharpening* avant de s'en servir sur des images satellite téléchargées sur le site de l'IGN. Pour cela nous avons bénéficié de l'aide d'un ingénieur de chez Thales qui connaissait particulièrement le sujet et qui a pu nous orienter vers différentes méthodes.

2.2 Objectifs techniques

Définissons les critères de qualités et les méthodes de traitement d'images employées.

2.2.1 Les critères de qualité

Le contraste

Le contraste est défini de la manière suivante :

Le contraste est une propriété intrinsèque d'une image qui quantifie la différence de clarté [1] entre les parties claires et sombres.

On parle aussi de brillance, qui, en revanche, n'admet pas de définition technique. On illustre facilement la définition avec un simple exemple :

Figure 2.1 – Photos avec plus (à gauche) ou moins (à droite) de contraste.

Pour évaluer la saturation d'une image il est nécessaire de comparer les différents canaux de couleur de l'image (les fameux canaux RVB). En effet, les images en nuance de gris ont autant de rouge, que de vert, que de bleu pour chaque pixel ; dans ce cas l'image n'est pas du tout saturée. Au contraire, une image saturée verra les composantes de couleur de chaque pixel être très différentes.

2.2.2 Évaluer la qualité des images, différentes utilisations possibles

Selon le cahier des charges fourni par le client demandeur de l'image, une image satellitaire doit respecter différents critères. Nous nous sommes concentrés sur deux utilisations possibles de l'image :

— Premier cas : le client désire réaliser un **travail de précision** à partir de l'image (cartographie, analyse de terrain, etc...). Sa principale préoccupation sera la précision de image par rapport à la réalité (image non déformé avec des couleurs représentatives de la réalité).

— Deuxième cas : le client préfère obtenir un **produit esthétique** (posters, publicités, etc...). Le critère réaliste deviendra alors secondaire au profit d'autres critères visuels tels que le contraste, luminosité et la netteté.

2.3 Evaluation de la qualité des images

2.3.1 Première utilisation : Objectif précision

Étudier la précision du produit fini (l'image fusionnée par le processus de *Pan-Sharpening*) a été fait en le comparant à l'image panchromatique [2] du satellite (la plus précise disponible). Nous étudierons ici une vue de la ville du Havre obtenue à partir d'une certaine méthode de Pan-Sharpening, à l'aide de différents algorithmes de traitement et de Python. Afin de rendre comparables les deux produits, il est nécessaire de convertir l'image couleurs fusionnée en image de niveaux de gris.

Conversion de l'image en niveau de gris ou désaturation

L'algorithme de désaturation réalise une moyenne modérée des trois canaux couleur de l'image pour former une nouvelle image ayant un unique canal de luminance. La norme préférée pour la conversion sera la recommandation 709 qui réalise la moyenne pondérée suivante :

1. On parle de clarté et non pas de luminosité ; la différence sera établie plus loin.
2. On appelle panchromatique une réponse physico-chimique (émulsion photographique, pigment photosensible, etc.) qui ne discrimine pas les couleurs, c'est-à-dire dont la sensibilité à la longueur d'onde de la lumière est similaire à celle de la vision humaine.

$$Luminance = 0.2126 \times Rouge + 0.7152 \times Vert + 0.0722 \times Bleu$$

La formule étant appliquée à chaque pixel de l'image, on obtient une image de la forme numérique d'une matrice de dimension un de taille ($Hauteur \times Largeur$) :

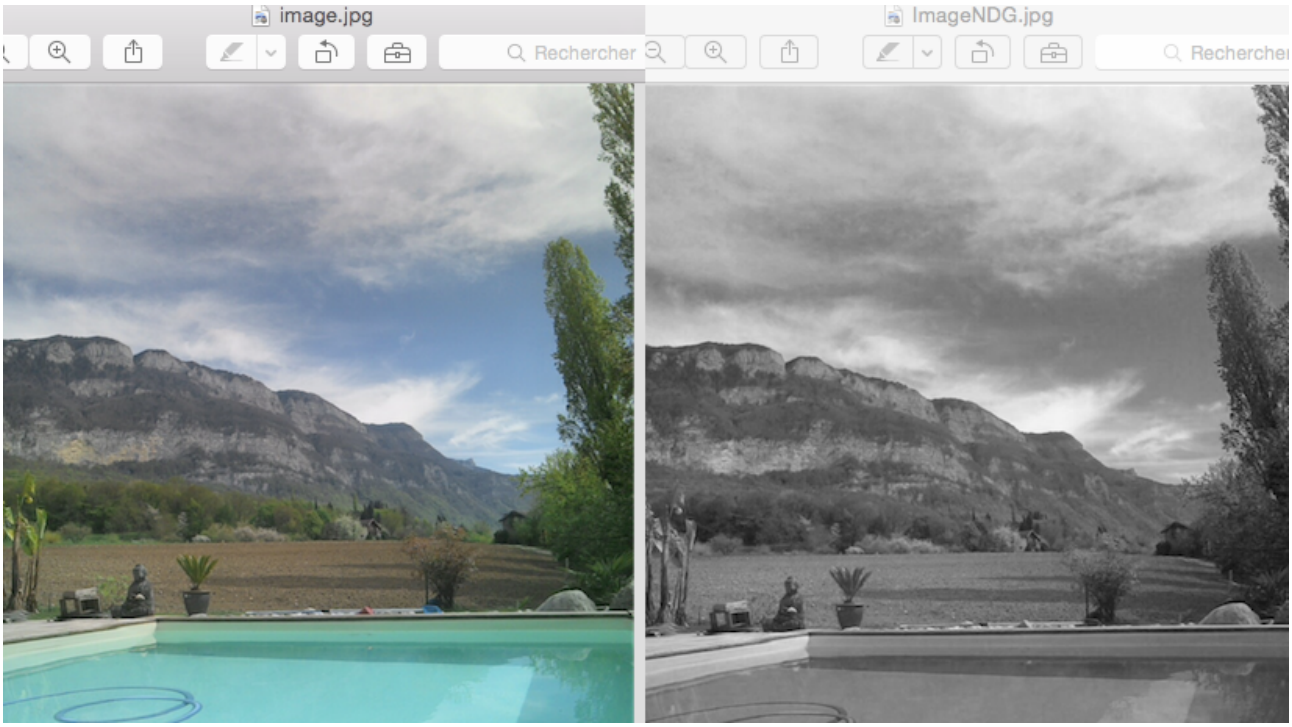

Figure 2.2 – Désaturation effectuée par l'algorithme

Première phase de traitement : Extraire les contours de l'image

Déterminer si l'image est proche d'une certaine vérité terrain demande de s'affranchir des couleurs, pouvant interférer avec l'analyse de l'image par le processus d'évaluation automatique utilisé. Ce qui nous intéressera dans l'image sera l'analyse des contours soit, des différentes formes géométriques observables.

Filtre de Sobel Parmi les différentes méthodes mathématiques permettant d'extraire les contours d'une image, le filtre dit de Sobel [3] permet d'obtenir de manière systématique à partir d'une image en niveau de gris quelconque une image représentant les contours sous forme de lignes grises plus ou moins blanches selon la netteté du contour (le contour est-il marqué ou subtil).

Ce filtre, bien que relativement simple, permet d'obtenir des résultats corrects sur beaucoup d'images et, comme beaucoup de filtres de traitement, le filtre de Sobel utilise les produits de matrices de convolution.

Convolution et filtres Chaque filtre utilisant la convolution de matrices possède ses matrices propres. Cela consiste en l'association à chaque pixel d'une matrice généralement 3x3 décrivant son état et celui des 8 qui l'entourent. Dans le cas du filtre de Sobel, la matrice associera les niveaux d'intensité (plus ou moins clair) des pixels. La convolution intervient ici, la matrice correspondant à chaque pixel sera convolue par les matrices propres au filtre, c'est à dire que chaque nombre de la matrice soit multiplié

3. Le nom de ce filtre vient du nom de son inventeur : Irwin Sobel (né le 12 Septembre 1940) un chercheur dans le domaine des images digitales

Figure 2.3 – Exemple d'utilisation d'un filtre de Sobel (celui de Scipy)

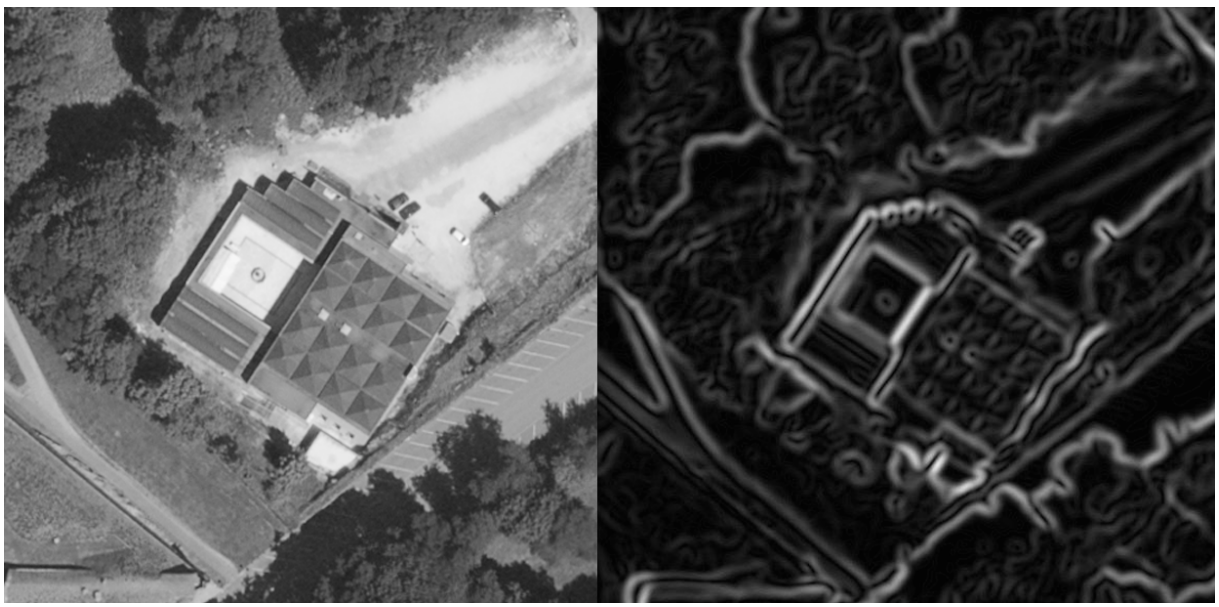

Figure 2.4 – Autre exemple d'utilisation d'un filtre de Sobel (celui de Scipy)

par celui correspondant dans l'autre matrice. Dans le cas du filtre de Sobel, les matrices suivantes seront concernées, avec A la matrice associée au pixel en cours de traitement :

$$G_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \star A$$

$$G_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \star A$$

Noter que $\star$ est ici le symbole du produit de convolution de matrices.
On associe finalement à A :

$$A = \sqrt{G_x^2 + G_y^2}$$

Où le carré désigne le produit de convolution de la matrice par elle même, de même pour la racine. Ceci forme l'image obtenue après filtrage pixel par pixel, A étant d'autant plus intense que la différence d'intensité entre deux pixels de l'image originale est élevée. La particularité du filtre de Sobel est qu'il traite séparément les voisinages vertical et horizontal du pixel, d'où la multiplication par deux matrices différentes.

Limites du filtre de Sobel et débruitage Malgré l'efficacité du filtre, certains signaux parasites peuvent venir dégrader le signal diminuant la précision de l'algorithme d'évaluation. Par exemple, le filtre à tendance à faire apparaître des contours inutiles et difficiles à traiter (arbres, pixels blancs ne représentant aucun contour...) :

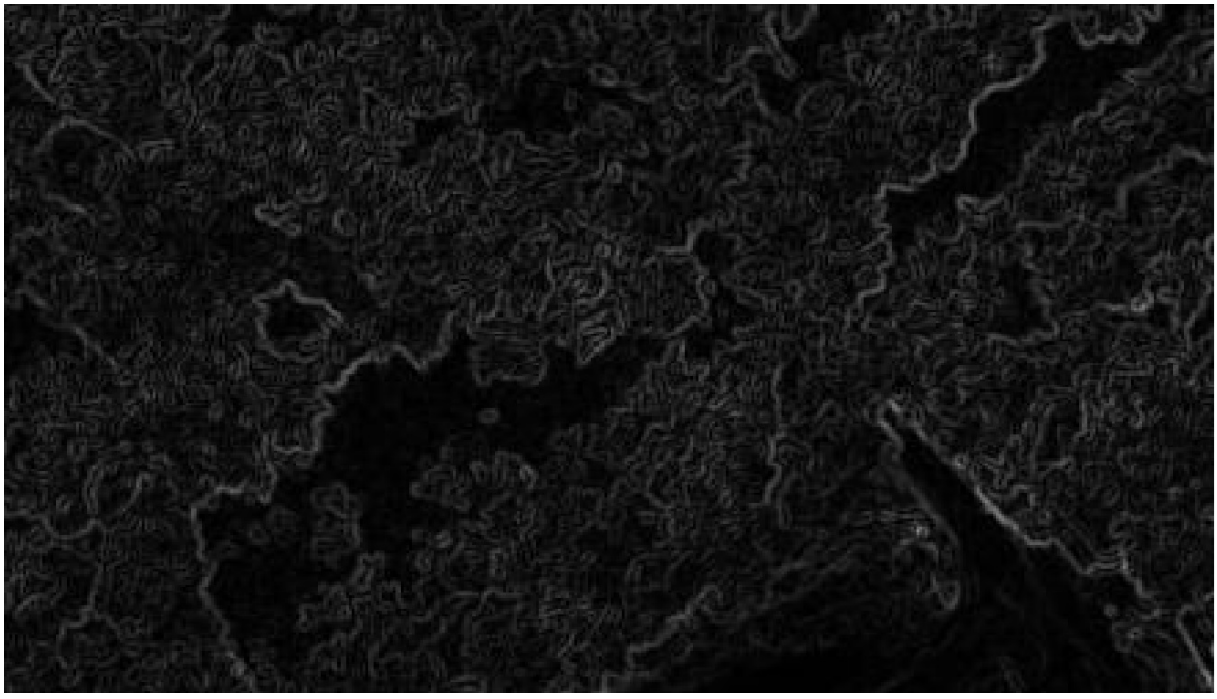

Figure 2.5 – Arbres sur une vue satellite filtrée de la ville de Belfort.

Ce genre de contours ont la particularité d'être d'intensité plus faible que les contours que l'on veut comparer. Ils constituent alors un bruit qu'il sera possible de d'éliminer à l'aide d'un filtre de seuillage qui affectera à chaque pixel la valeur 0 (noir) si celui-ci à une intensité inférieure à un certaine limite prédéfinie :

Figure 2.6 – Seuillage de 30. (Avant : gauche → après : droite)

Deuxième phase de traitement, comparaison et quantification

Après avoir appliqué le précédent processus aux images panchromatique et aux images fusionnée, on obtient deux fichiers images comparables. Evaluer la précision de l'image fusionnée s'effectuera par évaluation de l'erreur entre cette dernière et la bande panchromatique.

Le premier algorithme rentrant en jeu renvoie, à partir de la luminance des deux pixels correspondant sur les deux images, une valeur d'erreur relative de chaque pixel selon la formule suivante :

$$E_{x,y} = \left| \frac{(Luminance\ fusion)_{x,y} - (Luminance\ panchro)_{x,y}}{(Luminance\ panchro)_{x,y}} \right|$$

Les coordonnées x, y correspondent aux coordonnées du pixel sur les deux images. $E_{x,y}$ sera ensuite affecté à une matrice à la coordonnée x, y. Après le test de tous les pixels, l'algorithme aura créé une matrice d'erreur relative de dimension $(largeur\ image) \times (hauteur\ image)$.

Quantifier l'erreur relative totale de l'image sera finalement réalisé par moyenne quadratique ou RMS [4] ainsi que par la médiane.

Principe de la moyenne quadratique La moyenne quadratique est un outil mathématique permettant d'évaluer non pas une valeur moyenne comme pourrait le faire une moyenne arithmétique classique mais une valeur dite efficace, renseignant en plus les écarts de valeurs. Elle s'obtient par la formule suivante :

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}((x_i)^2)}$$

Dans notre cas, l'algorithme devra renvoyer, à partir de la matrice des erreurs relatives la valeur Pr suivante pour une image de dimension $L \times W$:

4. Root mean square

$$Pr = \sqrt{\frac{1}{(L+1)\times(W+1)}\sum_{x=0}^{L}(\sum_{y=0}^{W}(E_{xy})^2)}$$

Pourquoi la moyenne quadratique ? La moyenne quadratique est une moyenne dont la valeur varie d'autant plus qu'un nombre de valeurs analysées de même niveau est élévé. Cela sous entend que, contrairement à une moyenne arithmétique, les valeurs aberrantes (qui diffèrent grandement de la majorité des valeurs) seront très peu prises en compte dans la résultat final.

2.3.2 Deuxième utilisation : Objectif esthétique

Laissons désormais de coté le critère de vérité. L'esthétisme du produit fini (l'image fusionnée) sera évaluée non pas par comparaison de l'image fusionnée et de l'image panchromatique mais par évaluation des critères suivants sur l'image fusionnée seule :

- Contraste ;
- Luminosité ;
- Netteté.

Nous nous sommes concentrés ici sur la netteté et le contraste

Le contraste

Chapitre 3

Les ressources utilisées

Pour travailler sur notre projet nous avons eu besoin de rassembler différentes ressources. Premièrement nous avions besoin d'images satellites à exploiter, pour cela nous sommes allé sur le site internet de l'IGN qui proposait aux étudiant un accès à des images satellites avant leur publication.

3.1 Langage Python sous environnement Spyder

Spyder est un environnement [1] de développement facilitant la programmation en langage Python. Le langage Python est un langage interprété ; à la différence d'un langage compilé, il n'a pas besoin d'être transformé en fichier exécutable pour être utilisé : on le tape puis on lance l'interprétation du code. A chaque ligne de code l'interpréteur Python va agir en conséquence de ce qu'il lit ; il va par exemple mémoriser quelque chose, faire un test, commencer un boucle, afficher une donnée, en demander une autre à l'utilisateur, faire un calcule, etc... Chacune des lignes est prévue pour indiquer quoi faire à l'interpréteur. Au moindre doute du logiciel (à la moindre erreur dans le code) l'interpréter arrêtera le processus et renverra le message d'erreur correspondant à la ligne incomprise.

3.1.1 Python

D'après *Wikipédia* nous avons la description suivante :

> Python est un langage de programmation objet [2], multi-paradigme [3] et multiplateformes. Il favorise la programmation impérative structurée [4], fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort [5], d'une gestion automatique de la mémoire par ramasse-miettes [6] et d'un système de gestion d'exceptions ; il est ainsi similaire à Perl, Ruby, Scheme, Smalltalk et Tcl.

Les interpréteurs Python sont téléchargeable gratuitement en différentes versions, chaque version comprend son lot de module. Un module est un objet informatique permettant d'appliquer automatiquement différents algorithme pré-programmés par les personne ayant construit le module. Par exemple, le module **math** nous a servi pour tous les calcules sur les nombres, le module **matplotlib.pyplot** nous a servi à modifier et créer des images aussi bien en couleur qu'en nuance de gris, ainsi qu'à afficher des graphiques. Nous aurions aussi pu utilisé le module **PIL** [7] pour modifier les images ; mais il était

1. Espace de travail/logiciel permettant de programmer.
2. En Python tout est considéré comme un objet ; nombre, variable, liste,etc...
3. Permet l'exécution de plusieurs tâches en même temps
4. Impérative signifie que l'interpréteur exécute les ordre (lignes) qu'on lui donne ; structuré signifie qu'il le fait ligne par ligne.
5. Chacune des variable admet un type qui la caractérise : nombre entier, nombre à virgule, chaine de caractère, liste d'objet, tuple d'objet, dictionnaire, etc...
6. Processus qui permet de classer les taches à faire pour les exécuter dans un ordre optimal.
7. **PIL** est un module très puissant mais a été dépassé par le module **pillow**.

nécessaire de l'installer sur toutes les machine sur lesquelles nous allions travailler, et nous n'avions pas assez de temps. PIL signifie *Python Image Library*.

3.1.2 Spyder

D'après le site de Spyder nous avons la description (traduite) suivante :

> Spyder est un environnement de développement Python interactif fournir par MATLAB [8] avec de multiples fonctionnalités dans un logiciel simple et très léger. Il est fournit prête à l'emploi. C'est un éditeur de code-source avec coloration syntaxique, introspection, fonctions d'analyse, éditeur de tableau NumPy, éditeur de dictionnaire, console Python, etc...

D'après *Wikipédia* nous avons la description de spyder suivante :

> Spyder (anciennement dénommé Pydee) est un IDE [9] open source [10] multi-plateforme pour programmation scientifiques en langage Python. Spyder intègre par défaut les modules **NumPy**, **SciPy**, **Matplotlib** et **IPython** [11], ainsi que d'autres logiciels open source.
>
> En comparaison avec d'autres environnements de développement scientifique, Spyder a une unique combinaison de caractéristiques : multi-plateforme, open-source, écrit en Python et disponibles sous license non-copyleft [12]. Spyder est extensible avec des plugins [13], il est le support pour des outils interactifs, utile à l'analyse de données, il est l'assurance de la qualité d'un code spécifique à Python. Il est disponible sur différentes plateformes : Mac (sous le nomAnaconda), Windows (sous les noms WinPython et Python), et sur les principales distributions Linux comme Ubuntu, Debian, Fedora, openSUSE, Gentoo ou ArchLinux.

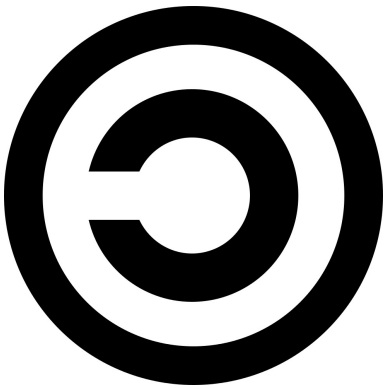

Figure 3.1 – Logo Copyleft.

3.2 Compilateur C++

Pour des besoins de test d'algorithmes téléchargés sur internet, nous avons eu besoin d'un compilateur C++. En effet, les fichiers que nous avons pu téléchargés sur *www.ipol.im* étaient tous écrits en C++. Il n'était évidement pas nécessaire d'un compilateur pour ouvrir les fichiers écrits en C++ (un simple éditeur

8. MATLAB est une filiale de logiciel informatique du développeur MathWorks qui signifie Matrix Laboratory ; MATLAB est alors un logiciel très puissant pour la gestion de système représentable par des matrices. MATLAB est aussi un langage informatique (tel que Python).

9. De l'anglais "Integrated Development Environment" : environnement de développement intégré.

10. Qui peut être modifié et distribué librement.

11. Différents modules permettant la gestion de données scientifiques. NumPy pour traiter les nombre, SciPy pour les outils scientifiques généraux, etc...

12. Copyleft est l'inverse de copyright. Copyleft n'est pas légalement reconnu mais est reconnu comme le refus par l'auteur de ses droit légitimes.

13. Ajouts au logiciel, tel que des nouveaux modules.

de texte suffisait), mais pour tester les algorithmes téléchargés nous avons eu besoin de les compiler ; d'ou l'intérêt du compilateur.

Le C++ est un langage de programmation qui nécessite une compilation avant de pouvoir interagir avec l'algorithme. Il est écrit dans une syntax bien plus complexe et primitive qu'en Python, cependant, son exécution est largement plus rapide : environ 20 fois plus rapide.

3.2.1 GCC

La compilation repose sur le principe de la transcription d'un scripte écrit en langage connu (script en C++), en un ensemble de données langage machine (*assembleur*[14]). GCC signifie : *GNU Compiler Collection*, il fait donc partie de la suite de logiciels développés initialement pour GNU : un projet de système d'exploitation libre lancé en 1983. On peut citer GIMP comme autre logiciel de la même famille. GIMP signifie *GNU Image Manipulation Program*, c'est un logiciel gratuit de modification d'image.

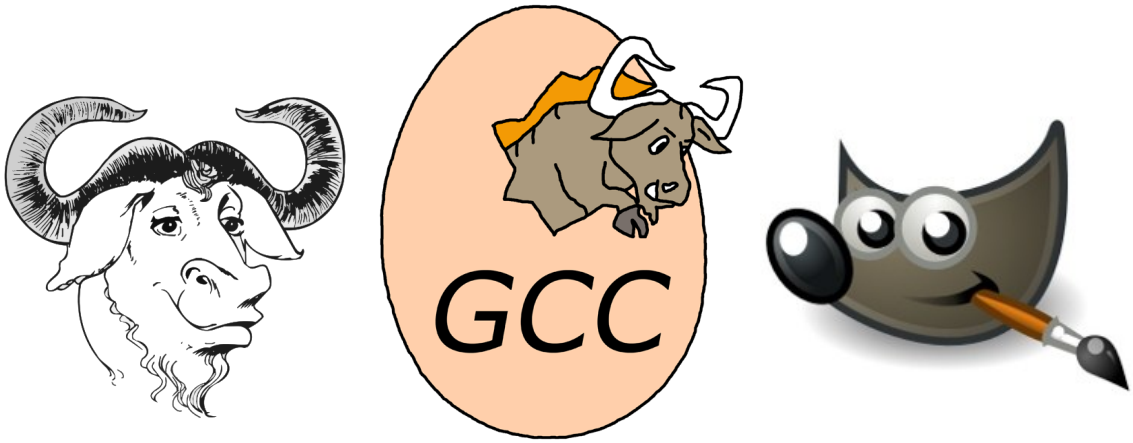

Figure 3.2 – Les logos (de gauche à droite) de GNU, GCC et GIMP

3.3 Nos fonctions

Pour traiter les images nous avons procédé par étape. Nous avons créer des fonctions informatiques permettant d'effectuer une étape précise (exemple : redimensionner l'image, dénaturer l'image, appliquer un certain filtre, etc...), puis nous avons mis à la suites ces différentes fonctions afin de constituer le corps de notre algorithme. Nous avons donc pu créer une multitude de fonction exécutant différentes tâches. Nous les détaillons ici, et nous en ferons appel dans la partie résultat afin d'expliquer les corps d'algorithme. (Nous définissons ici les multiples étapes possibles de l'algorithme, alors que nous ne décrirons que plus tard dans quel ordre nous avons décidé d'enchainer ces étapes.)

Dans la plupart des fonctions nous avons fait afficher le pourcentage d'execution de la fonction. Pour cela nous avons fait afficher le pourcentage de la largeur parcourue par l'algorithme.

14. Code extrêmement primitif permettant de visualiser un grand nombre de bits (0 ou 1) de manière condensée.

3.3.1 Redimensionnement

```
def redim(chemin_image_source,nom_du_fichier_final,pas):
    """Permet de rediomentinner des images de depart pour eviter d'avoir des
    dimentions tailles d'images differentes apres la degradation.

    SYNTAX:
    redim(chemin_image_source,nom_du_fichier_final,pas)
    """
    im=plt.imread(chemin_image_source)
    a=np.shape(im)
    hauteur=a[0]
    largeur=a[1]
    n=0
    new_largeur=(largeur//pas)*pas
    new_hauteur=(hauteur//pas)*pas
    MAT=np.zeros((new_hauteur,new_largeur,3))
    for x in range(new_largeur):
        n+=1
        print("Redim %largeur : ",(n/new_largeur)*100)
        for y in range (new_hauteur):
            MAT[y][x]=im[y][x]
    sc.misc.imsave(nom_du_fichier_final,MAT)
```

Cette fonction prend en argument :

- Le chemin d'accès de l'image à redimensionner.
- Le nom de la future image qui sera enregistrée.
- Le pas de la dégradation. (Nombre de pixel entre chaque pixel extrait lors de la dégradation). Ce pas de dégradation est utile a connaitre afin de redimensionner l'image de sorte a ce que ses dimensions soient un multiple du pas de dégradation.

Cette fonction ne retourne rien.

Par contre cette fonction a l'action suivantes :

- Elle enregistre l'image redismentionnée à l'emplacement courant [15] de l'algorithme.

15. Chemin d'accès correspondant au lieux d'exécution de l'interpréter Python. (Toute console (y compris les interpréter Python) s'exécutent obligatoirement à un lieu précis).

3.3.2 Degradation

```
1  def degrad (chemin_image_source,pas,nom_du_fichier_final):
2      """Permet de degrader
3      une image en ne prenant qu'un pixel sur [pas]
4      puis redimentionne l'image (modulo le pas) a la taille d'origine
5      avant de sauvegarder le fichier.
6
7      SYNTAX :
8      degrad (chemin_image_source,pas,nom_du_fichier_final)"""
9      im=plt.imread(chemin_image_source)
10     a=np.shape(im)
11     hauteur=a[0]
12     largeur=a[1]
13     n=0
14     MAT=np.zeros((hauteur//pas,largeur//pas))
15     for x in range(largeur//pas):
16         n+=1
17         print("Degradation %largeur : ",(n/(largeur)//pas)*100)
18         for y in range (hauteur//pas):
19             MAT[y][x]=im[y*pas][x*pas]
20     final=scipy.ndimage.zoom(MAT,pas)
21     sc.misc.imsave(nom_du_fichier_final, final)
```

Cette fonction prend en argument :

- Le chemin d'accès de l'image à dégrader.
- Le nom de la future image qui sera enregistrée.
- Le pas de la dégradation. (Nombre de pixel entre chaque pixel extrait).

Cette fonction ne retourne rien.

Par contre cette fonction a l'action suivantes :

- Elle enregistre l'image dégradée à l'emplacement courant [16] de l'algorithme.

3.3.3 Filtre de Sobel

Le filtre de Sobel est un processus permettant de prendre une image en nuance de gris en entree, et de retourner une nouvelle image en nuance de gris où les contours presents dans l'image d'entrée sont représentés par des traits plus ou moins blanc selon la dureté du contour. En absence de contour, l'image de sortie sera intégralement noire.

16. Chemin d'accès correspondant au lieux d'exécution de l'interpréter Python. (Toute console (y compris les interpréter Python) s'exécutent obligatoirement à un lieu précis).

```
def sobel (chemin_image_source,nom_du_fichier_final):
    """Applique un filtre de sobel a une image
    avant de la sauvegarder dans le repertoire courant.

    SYNTAX :
    sobel (chemin_image_source,nom_du_fichier_final):
    nom_du_fichier_final comprendra l'extension.
    """
    im = sc.misc.imread(chemin_image_source)
    im = im.astype('int32')
    dx = ndimage.sobel(im, 0)  # horizontal derivative
    dy = ndimage.sobel(im, 1)  # vertical derivative
    mag = np.hypot(dx, dy)  # magnitude
    mag *= 255.0 / np.max(mag)  # normalize (Q&D)
    sc.misc.imsave(nom_du_fichier_final, mag)
```

Cette fonction prend en argument :

- Le chemin d'accès de l'image à filtrer.
- Le nom de la future image qui sera enregistrée.

Cette fonction ne retourne rien.
Par contre cette fonction a l'action suivantes :

- Elle enregistre l'image une fois le filtre de Sobel appliqué à l'emplacement courant de l'algorithme.

3.3.4 Seuillage

Le seuillage est un processus permettant de prendre une image en nuance de gris en entrée, et de retourner une nouvelle image en nuance de gris où les pixel ayant un luminance au dessous d'un seuil sont remplacés par un pixel noir.

```
def seuil (nseuil,chemin_image_source,nom_du_fichier_final):
    """Aplique un seuillage a une image
    avant de la sauvegarder dans le repertoire courant.

    SYNTAX :
    seuil (seuil,chemin_image_source):
    nom_du_fichier_final comprendra l'extension.
    """
    im=sc.misc.imread(chemin_image_source)
    a=np.shape(im)
    hauteur=a[0]
    largeur=a[1]
    n=0
    for x in range(largeur):
        n+=1
        print("Seuil %largeur : ",(n/largeur)*100)
        for y in range (hauteur):
            if im[y][x]<nseuil:
                im[y][x]=0
    sc.misc.imsave(nom_du_fichier_final,im)
```

Cette fonction prend en argument :

- Le seuil de la dégradation (nombre compris entre 0 et 255).

- Le chemin d'accès de l'image à seuiller.
- Le nom de la future image qui sera enregistrée.

Cette fonction ne retourne rien.
Par contre cette fonction a l'action suivantes :

- Elle enregistre l'image une fois seuillée à l'emplacement courant de l'algorithme.

3.3.5 Fermeture

```
1 def fermeture(chemin_image_source,nom_du_fichier_final,size):
2     im=plt.imread(chemin_image_source)
3     imferme=scipy.ndimage.morphology.grey_closing(im,size)
4     sc.misc.imsave(nom_du_fichier_final,imferme)
```

Cette fonction prend en argument :

- Le chemin d'accès de l'image dont on veut fermer les contours.
- Le nom de la future image qui sera enregistrée.
- Un tuple correspondant à la taille de la fenêtre de convolution nécessaire à la fermeture comprenant :
  - La largeur de la fenêtre en première position (0).
  - La hauteur de la fenêtre en deuxième position (1).

Cette fonction ne retourne rien.
Par contre cette fonction a l'action suivantes :

- Elle enregistre l'image une fois ses contours fermés à l'emplacement courant de l'algorithme.

3.3.6 Matrice d'erreur relative

```
def matriceec(chemin_image_1,chemin_image_2):
    """Retourne la matrice de difference relative entre deux images de meme taille.

    SYNTAX :
    matriceec (chemin_image_1,chemin_image_2):
    """
    im1=plt.imread(chemin_image_1)
    im2=plt.imread(chemin_image_2)

    a=np.shape(im1)
    hauteur=a[0]
    largeur=a[1]
    MAT=np.zeros((hauteur,largeur))
    n=0
    if a!=np.shape(im2):
        return 'error dim'
    else:
        for x in range(largeur):
            n+=1
            print("Erreur relative %largeur : ",(n/largeur)*100)
            for y in range (hauteur):
                if im2[y][x]==0:
                    pass
                elif im1[y][x]!=0 :
                    MAT[y][x]=(abs((im2[y][x]-im1[y][x])/im1[y][x]))
                else :
                    MAT[y][x]=(abs((im2[y][x]-im1[y][x])))
    N=hauteur*largeur
    return (MAT,N)
```

Cette fonction prend en argument :

- Le chemin d'accès d'une première image (Image1).
- Le chemin d'accès d'une deuxième image (Image2).

Cette fonction retourne :

- Un tuple de résultat qui contient :
  - La matrice d'erreur en première position (0).
  - La nombre de pixel effectivement pris en considération dans le calcul en deuxième position (1).

3.3.7 Moyenne RMS

```
def RMS(MAT,N):
    """Retourne la moyenne RMS de difference entre deux image
    en entrant la matrice de difference.

    SYNTAX :
    RMS(MAT,N)
    """
    somme=0
    a=np.shape(MAT)
    hauteur=a[0]
    largeur=a[1]
    n=0
    for x in range(largeur):
        n+=1
        print("RMS %largeur : ",(n/largeur)*100)
        for y in range (hauteur):
            somme+=MAT[y][x]**2
    return (sqrt((1/N)*somme))
```

Cette fonction retourne :

-
-
-
-

De plus cette fonction a les action suivantes :

-
-
-
-

3.3.8 Contraste de Michelson

```
def Michelson(chemin_image_source,h,l):
    """Permet de crer une matrice de Michelson : matrice qui contient pour
    chaque pixel un nombre representatif de l'homogeneite de ses voisins.

    SYNTAX :
    Michelson(chemin_image_source,h,l)
    h,l-->hauteur et largeur de la fenettre des voisins.
    """
    if h%2==0:
        raise Exception("h doit etre impair !")
    if l%2==0:
        raise Exception("l doit etre impair !")
    im=plt.imread(chemin_image_source)
    a=np.shape(im)
    n=a[0]
    prob=0
    p=a[1]
    N=0
    MAT=np.zeros((n-h+1,p-l+1))
    for x in range(p-l+1):
        N+=1
        print("Matrice contraste de Michelson %largeur : ",(N*1000//(p-l+1))/10)
        for y in range (n-h+1):
            liste=[]
            for xi in range(x,x+l):
                for yi in range(y,y+h):
                    liste.append(im[yi][xi])
            if (max(liste)+min(liste))==0:
                valeur=0
            elif (max(liste)-min(liste))/(max(liste)+min(liste))>1:
                prob+=1
                valeur=0
            else:
                valeur=(max(liste)-min(liste))/(max(liste)+min(liste))
            MAT[y][x]=valeur
    return (MAT,((((n-h+1)*(p-l+1)))-prob))
```

Cette fonction retourne :

- Un tuple de résultat qui contient :
  - La matrice de contraste de Michelson en première position (0).
  - Le nombre de pixels pris effectivement [17] en compte par cette matrice en deuxième position (1).

17.

Chapitre 4

Analyse des données

Cette première année les analyses de données auront été réalisé grâce à une mise en situation fictive. Nous avons simulé l'obtention d'une image couleur basse définition en dégradant volontairement une image haute définition. Nous avons analysé les images de trois manière :

- En attribuant un **score de similitude** de l'image dégradée à l'image haute définition ;
- En attribuant un **score de contraste** à l'image haute définition ;
- En attribuant un **score de netteté** à l'image haute définition.

La syntax utilisé pour le stockage de nos images a été la suivante :

[derniere modification] [programme utilise] sur [image avant la modification]

Exemple :

Redim fois4 python sur Sobel python degrade 4px python NDG photoshop

[derniere modification] : Redim fois4

[programme utilise] : python

[image avant la modification] : Sobel python degrade 4px python NDG photoshop

4.1 Score de similitude

Pour répondre à notre problème de quantification de la fidélité de l'image au terrain nous avons dont comparé nos deux images fictives décrites plus haut. C'est l'algorithme ci dessous qui a été utilisé pour comparer les deux images. Il fait appel à de nombreuses fonctions décrites dans le chapitre des ressources utilisées.

```
##########################################
#Algorithme fictif appelant les fonctions.
##########################################

def evaluationRMSfic(chemin_image,chemin_dossier_de_travail,size,pas=4,nseuil=100,R=0.2126,
    V=0.7152,B=0.0722):
    """Algorithme permettant de renvoyer une moyenne RMS des
    differences entre une image en nuance de gris (NDG) et une image
    en couleur.
    Cette moyenne RMS est un nombre representatif de la fidelite entre
    les deux images.

    SYNTAX :
    evaluationRMS(chemin_image1_NDG,chemin_image2_couleur,chemin_dossier_de_travail)
    [chemin_dossier_de_travail]-->Dossier ou les images temporaires seront stockees.
    [pas]-->pas en pixel utilise lors de la degradation.
    [seuil]-->seuil de 0 a 255 utilise pour enleve les parasite apres fitrage Sobel.
    [R], [V], et [B] sont des nombres les coffeicient d'importancce des couleurs.
    1 veut dire que l'importance est normale"""
    import os
    os.chdir(chemin_dossier_de_travail)

    ####################
    #Redimentionnement 1 et 2
    #redim(chemin_image_source,nom_du_fichier_final,pas)
    redim(chemin_image,"Image1_HD_redim.tif",pas)
    redim(chemin_image,"Image2_HD_redim.tif",pas)
    print("Redimensionnement fini")

    ####################
    #Desaturation de 1,2
    #desaturation(chemin_image_source,nom_du_fichier_final,R=1,V=1,B=1)
    desaturation("Image1_HD_redim.tif","Image1_HD_NDG_redim.tif",R=1,V=1,B=1)
    desaturation("Image2_HD_redim.tif","Image2_HD_NDG_redim.tif",R=1,V=1,B=1)
    print("Desaturation finie")

    ####################
    #Degradation de 1
    #degrad (chemin_image_source,pas,nom_du_fichier_final)
    degrad ("Image2_HD_NDG_redim.tif",pas,"Image2_degrade_HD_NDG_redim.tif")
    print("Degardation et redimensionnement finis")

    ####################
    #Filtres de Sobel
    #sobel (chemin_image_source,nom_du_fichier_final)
    sobel ("Image1_HD_NDG_redim.tif","Image1_sobelisee_HD_NDG_redim.tif")
    sobel ("Image2_degrade_HD_NDG_redim.tif","Image2_sobelisee_degrade_HD_NDG_redim.tif")
    print("Filtre Sobel fini")

    ####################
    #Seuillage
    #seuil ( seuil ,chemin_image_source,nom_du_fichier_final)
    seuil (nseuil,"Image1_sobelisee_HD_NDG_redim.tif","
        Image1_seuillee_sobelisee_HD_NDG_redim.tif")
    seuil (nseuil,"Image2_sobelisee_degrade_HD_NDG_redim.tif","
        Image2_seuillee_sobelisee_degrade_HD_NDG_redim.tif")
    print("Seuillage fini")
```

```
    ####################
    #Fermeture
    fermeture("Image1_seuillee_sobelisee_HD_NDG_redim.tif","
        Image1_ferme_seuillee_sobelisee_HD_NDG_redim.tif",size)
    fermeture("Image2_seuillee_sobelisee_degrade_HD_NDG_redim.tif","
        Image2_ferme_seuillee_sobelisee_degrade_HD_NDG_redim.tif",size)

    ####################
    #Matrice d'erreur
    #matriceec (chemin_image_1,chemin_image_2)
    MATRICE_ERREUR=matriceec ("Image1_ferme_seuillee_sobelisee_HD_NDG_redim.tif","
        Image2_ferme_seuillee_sobelisee_degrade_HD_NDG_redim.tif")
    print("Matrice d'erreur finie")

    ####################
    #RMS
    #RMS(MAT,N)
    RESULTAT=RMS(MATRICE_ERREUR[0],MATRICE_ERREUR[1])

    ####################
    #Stockage fichier resultat (chose inoperante)
    fichier=open("Resultat","w")
    fichier.write("RMS;Moyenne;Min;Max;Pas de degradation (pixel);size[0];size[1];Seuil (inf
        255);R;V;B\n")
    fichier.write("%s;%s;%s;%s;%s;%s;%s;%s;%s;%s;%s;%s\n"%(RESULTAT,np.mean(MATRICE_ERREUR[0]),
        np.min(MATRICE_ERREUR[0]),np.max(MATRICE_ERREUR[0]),pas,size[0],size[1],nseuil,R,V,B
        ))
    fichier.close()

    sc.misc.imsave("MATRICE_ERREUR.jpg",MATRICE_ERREUR[0])
    return RESULTAT,np.mean(MATRICE_ERREUR[0]),np.min(MATRICE_ERREUR[0],np.max(
        MATRICE_ERREUR[0]))
```

Ce programme utilise les fonctions définies dans le chapitre des ressources utilisées et renvoie en sortie les éléments suivants (dans l'ordre) :

- La moyenne RMS des des valeurs de différences entre les deux images ;
- La moyenne arithmétique des valeurs de différences entre les deux images ;
- La minimum des valeurs de différences entre les deux images ;
- La maximum des valeurs de différences entre les deux images.

De plus, l'algorithme créer un fichier texte et stocke les résultats à l'intérieur afin de permettre une future analyse de donnée (l'année prochaine) réalisée par comparaison des images réelles du satellite. Nous pourrons alors exploiter les données de ce fichier texte pour tracer des courbes, des diagrammes, etc...

4.2 Score de contraste

```
##########################################
#Algorithme score de contraste
##########################################

def Contraste(chemin_image,chemin_dossier_de_travail,h=3,l=3):
    """Algorithme permettant de renvoyer une moyenne RMS de la quantite de
    contours d'une image. On appelera la valeur obtenue le "score de contraste".
    PREND UNE IMAGE COULEUR EN ENTREE

    SYNTAX :
    Contraste(chemin_image,chemin_dossier_de_travail,h=3,l=3)
    [chemin_dossier_de_travail]-->Dossier ou les images temporaires seront stockees.
    [h], et [l] sont dimension de la fenettre de convolution (en pixel).
    """
    import os
    os.chdir(chemin_dossier_de_travail)

    ######################1
    #Passage en niveau de gris
    #desaturation(chemin_image_source,nom_du_fichier_final,R,V,B)
    desaturation(chemin_image,"Image_NDG.tif",R=0.2126,V=0.7152,B=0.0722)

    ######################2
    #Matrice de Michelson
    #Michelson(chemin_image_source,h,l)
    MATRICE_MICHELSON=Michelson("Image_NDG.tif",h,l)

    ######################3
    #RMS
    #RMS(MAT,N)
    RESULTAT=RMS(MATRICE_MICHELSON[0],MATRICE_MICHELSON[1])

    return RESULTAT
```

4.3 Score de netteté

```
def nettete(chemin_image,chemin_dossier_de_travail):
    """Evalue la nettete de l'image testee.
    PREND UNE IMAGE COULEUR EN ENTREE

    SYNTAX :
    nettete (chemin_image,chemin_dossier_de_travail,nseuil)
    nom_du_fichier_final comprendra l'extension.
    """
    import os
    os.chdir(chemin_dossier_de_travail)

    ######################1
    #Passage en niveu de gris
    #desaturation(chemin_image_source,nom_du_fichier_final,R,V,B)
    desaturation(chemin_image,"Image_NDG.tif",R=0.2126,V=0.7152,B=0.0722)

    ######################2
    #Filtres de Sobel
    #sobel (chemin_image_source,nom_du_fichier_final)
    sobel("Image_NDG.tif","Image_sobelisee_NDG.tif")

    ######################3
    #RMS
    #RMS(MAT,N)
    MAT=sc.misc.imread("Image_sobelisee_NDG.tif")
    a=np.shape(MAT)
    RESULTAT=RMS(MAT,a[0]*a[1])

    return RESULTAT
```

Chapitre 5

Perspectives

De la même manière, la réflexion sur les perspectives et sur les applications pratiques sera réalisée en deuxième année.

Table des figures

Liste des tableaux