

Rapport d'analyse

Cycle des Ingénieurs de l'ENSG 3^{ème} année Spécialité PPMD

Création d'une boîte à outils de Tone Mapping en python



FIGURE 1 – Image HDR [1]

Amaury Blotin
Commanditaire : Manchun Lei

04 décembre 2024

Non confidentiel Confidential IGN Confidential Industrie

ECOLE NATIONALE DES SCIENCES GÉOGRAPHIQUES
6-8 Avenue Blaise Pascal - Cité Descartes - 77420 Champs-sur-Marne
Téléphone 01 64 15 31 00 Télécopie 01 64 15 31 07

Table des matières

1 Présentation du sujet	2
1.1 Problématique	2
1.2 Entreprise et commanditaire	2
1.3 Objectifs	2
2 Expression du besoin	3
2.1 Les objectifs de l'étude	3
2.2 Contraintes à respecter	3
2.3 Cas d'utilisation	3
3 Analyse fonctionnelle	4
3.1 Exemple de TMO : Drago	4
3.2 Schématisation d'un traitement	5
4 Etude technique	6
4.1 Choix techniques	6
4.2 Organisation du package	8
4.3 Résultats obtenus	8
5 Gestion de projet	10
5.1 Les outils de suivi	10
5.2 Les risques	10
5.3 Le déroulement	11
5.4 Les perspectives	12
Références	12

1 Présentation du sujet

1.1 Problématique

En télédétection, il est intéressant de comparer des images aériennes en vraie couleur (TCI) et en infrarouge (IRC) à des images multispectrales satellitaires. Les images aériennes sont souvent en SDR (Standard Dynamic Range) et les images multispectrales en HDR (High Dynamic Range). Pour optimiser la comparaison entre ces deux types de prises de vues, il faut harmoniser la radiométrie de ces images. Cela passe par une compression de la plage dynamique de l'image multispectrale. Cette transformation se fait par l'utilisation d'un TMO (Tone Mapping Operator). De nombreux TMO existent, ils diffèrent par leur efficacité et leur capacité à conserver une grande quantité d'information. Ainsi on obtient une image multispectrale réduite en SDR.

Sur python, il n'y a pas de bibliothèque python libre permettant de gérer l'ensemble des étapes de modification d'une image HDR avec un choix varié de TMO, quel que soit le format de l'image tout en gardant le géoréférencement s'il est présent.

1.2 Entreprise et commanditaire

L'IGN (institut national de l'information géographique et forestière) présente une branche de recherche en imagerie. Monsieur Manchun Lei (mon commanditaire) est chargé de recherche à l'IGN et aimeraient avoir à disposition une boîte à outils comblant ce vide. En parallèle de mon projet, Manchun Lei commandite également un groupe dont l'objectif est de comparer les différents algorithmes de TMO pour en extraire les algorithmes les plus utiles à ses besoins.

1.3 Objectifs

Ce projet a pour but la création d'un package de fonctions implémentées pouvant être appelées dans un script pour manipuler les images HDR. Les premiers besoins énoncés par Manchun Lei ont été :

- Lire des images multispectrales géoréférencées
- Convertir les images vers une réponse radiométrique selon le standard CIE XYZ ou d'autres capteurs via des transformations matricielles
- Appliquer des algorithmes de tone mapping (TMOs) pour convertir les images HDR en SDR
- Sauvegarder les images SDR géoréférencées

En discutant davantage du projet, nous avons également identifié un autre besoin. Changer la plage dynamique d'une image va immanquablement changer le NDVI. Créer des TMO inverse lorsque cela est possible. Ainsi, dans le cas idéal, on pourrait calculer des indicateurs comme le NDVI, fidèles à l'image originale même si un TMO a déjà été appliqué.

2 Expression du besoin

2.1 Les objectifs de l'étude

Cette étude a pour but de cadrer le développement d'une boîte à outils et d'étudier le résultat. Ce projet fait interagir de nombreuses fonctions. Il était donc très important d'organiser clairement les fonctions.

2.2 Contraintes à respecter

Sur le plan technique, il y a quelques contraintes liées à l'utilisation qui sera faite du code.

-Les fonctions implémentées doivent l'être en python. Ce choix est motivé par le fait que le traitement d'image à l'IGN passe principalement par python. Ces fonctions pourront être utilisées dans une chaîne de traitement plus globale.

-L'utilisateur doit pouvoir choisir de garder des paramètres par défaut. Il peut dans le cas inverse avoir la main sur la totalité des paramètres d'un TMO.

-L'utilisateur doit pouvoir appliquer plusieurs traitements successifs sans avoir à ouvrir de nouveau une image. Il doit donc y avoir une image temporaire que l'on pourra choisir de sauvegarder après.

-Certaines images ne suivent pas le système colorimétrique CIE XYZ (Rouge, Vert, Bleu). Il existe d'autres systèmes tel que le HSV (Hue Saturation Brightness) ou le L*a*b (L pour la clarté, a pour la valeur sur l'axe vert-rouge et b celle sur l'axe bleu-jaune). Le système colorimétrique des images en sortie doit être le CIE XYZ. Cette contrainte est finalement gérée par une autre toolbox en amont.

2.3 Cas d'utilisation

Les trois actions principales que l'utilisateur sera amené à faire sont : l'ouverture d'une image, sa modification par un TMO et sa sauvegarde. D'autres outils pourront être utilisés séparément tels que ceux utilisés pour la conversion de système colorimétrique, les calculs de luminance et autres.

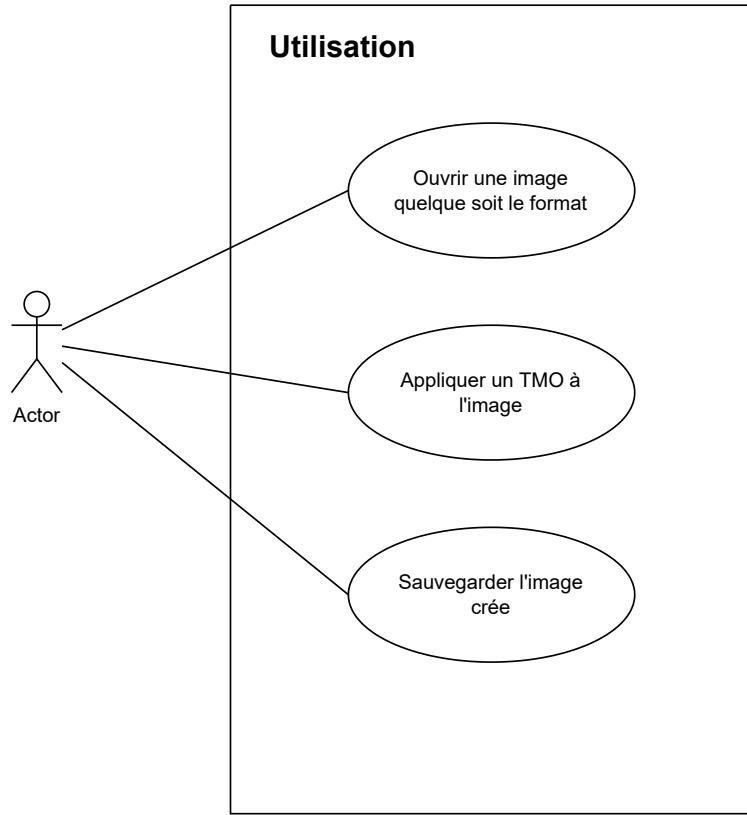


FIGURE 2 – diagramme d'utilisation

3 Analyse fonctionnelle

3.1 Exemple de TMO : Drago

Pour comprendre comment construire les fonctions prenons l'exemple du TMO Drago. Ce TMO utilise la Luminance de chaque pixel pour et lui applique une transformation à l'aide de différents paramètres. Cette fonction intègre un biais. Cette fonction ci-dessous, comme toutes celles décrites dans cette partie proviennent de l'article sur le TMO dit de Drago [1] .

$$\text{bias}_b(t) = t^{\frac{\log(b)}{\log(0.5)}}$$

où :

- t est l'intensité normalisée ($0 \leq t \leq 1$).
- b est le paramètre de biais ($0 < b \leq 1$).

Le biais ajuste l'intensité t en appliquant une transformation exponentielle contrôlée par le paramètre b . b permet d'ajuster la compression des valeurs hautes et la visibilité des

détails dans les zones sombres. Pour faire apparaître le plus de détails, il est conseillé de prendre b tel que $(0.7 < b \leq 0.9)$.

Pour calculer les nouvelles couleurs on utilise la fonction suivante :

$$Ld = \frac{L_{dmax} * 0.01}{\log_{10}(L_{w_{max}} + 1)} * \frac{\log(L_w + 1)}{\log(2 + ((\frac{L_w}{L_{w_{max}}})^{\frac{\log(b)}{\log(0.5)}}) * 8)}$$

- Ld : displayed Luminance - Luminance calculée
- Lw : world Luminance - Luminance d'un pixel
- L_{max} : maximum Luminance - Luminance maximal sur l'image
- L_{dmax} : maximum displayed Luminance - utilisé comme facteur d'échelle, valeur que l'on choisit comme maximum arbitraire. Généralement, la valeur $L_{dmax} = 100$ cd/m² est utilisée.

Suite à cela, il est encore nécessaire d'appliquer la correction gamma pour compenser la non linéarité du système d'affichage. On modifiera la Luminance avec la décomposition ci-dessous.

Cas 1 $L > start$: $L * slope$

Cas 2 $L < start$: $L * pow(0.9/gamma) - 0.099$

- $slope$: slope - tangente à la courbe au point origine de la fonction de correction gamma

$$Ld = Lw^{\frac{1}{gamma}}$$

- $gamma$: coefficient gamma, généralement gamma = 2.2

On a donc une chaîne de traitements dont plusieurs des paramètres sont arbitraires et peuvent varier (gamma, L_{dmax} , b). Ces paramètres auront par défaut des valeurs arbitraires correspondant aux valeurs conseillées. L'utilisateur pourra les changer s'il le souhaite.

L'algorithme Drago

3.2 Schématisation d'un traitement

Avant l'utilisation d'un TMO, il faut ouvrir l'image avec la classe Open, quel que soit son format (tiff, Geotiff, jpeg, png) et en récupérer une matrice modifiable avec les métadonnées importantes. Les métadonnées qui sont une pièce importante du projet sont enregistrés dans un fichier txt créé à l'exécution de la fonction openGDAL. Ce fichier ainsi

crée doit conserver les métadonnées de départ mais aussi garder une trace de chaque algorithme de TMO appliqué et les paramètres de celui-ci. Puis on peut lancer l'algorithme de TMO (DragoTMO expliqué plus haut ou GammaTMO dans le package). Cet algorithme peut faire appel à d'autres classes pour des opérations annexes. L'enregistrement du TMO utilisé et des paramètres dans les métadonnées se base sur l'utilisation d'un outil 'modify_metadata' par exemple. Puis on peut utiliser la classe Save pour enregistrer la nouvelle image et y intégrer le fichier de métadonnées.

Drago n'est finalement pas l'algorithme que j'ai implémenté. J'ai choisi en accord avec mon commanditaire de coder en premier TMO la correction Gamma qui a l'avantage d'être réversible et donc de permettre d'enchaîner l'ensemble des traitements jusqu'à revenir à une image qui doit ressembler à l'originale.

4 Etude technique

4.1 Choix techniques

A terme, cette Toolbox devrait être utilisée pour créer une plugin QGIS. L'extension QGIS Plugin Builder permet d'adapter un code python en plugin. Pour ce faire, il crée tout un dossier incluant un sous-dossier laissant de la place pour un package python. Pour faciliter cette étape, j'ai décidé d'organiser ce projet comme un package. Ainsi, lorsque le développement de la ToolBox sera fini, la création de plugin devrait se faire sans trop de difficulté.

Plutôt que de coder de manière procédurale, j'ai choisi de coder en Programmation Orientée Objet. Ce choix est motivé par le besoin de clarté dans la construction du projet pour permettre des évolutions apportées par d'autres développeurs. Grâce à la POO, l'utilisation des fonctions sera claire pour simple et intuitif. Ce système est utilisé par d'autres bibliothèques connues : Pandas, Tensorflow, Matplotlib...

Des librairies, telles que Pillow, GDAL ont été utilisées pour ouvrir modifier et enregistrer des images et leurs métadonnées. GDAL est une librairie permettant d'ouvrir la majorité des types de fichiers mais surtout, GDAL est particulièrement utile pour traiter les fichiers .tif géoréférencés. GDAL est parfois compliquée à installer du fait de litiges entre les versions de GDAL et d'autres bibliothèques qui se croisent. Pour mener à bien ce projet j'ai donc commencé par créer un environnement vide pour y installer en premier GDAL.

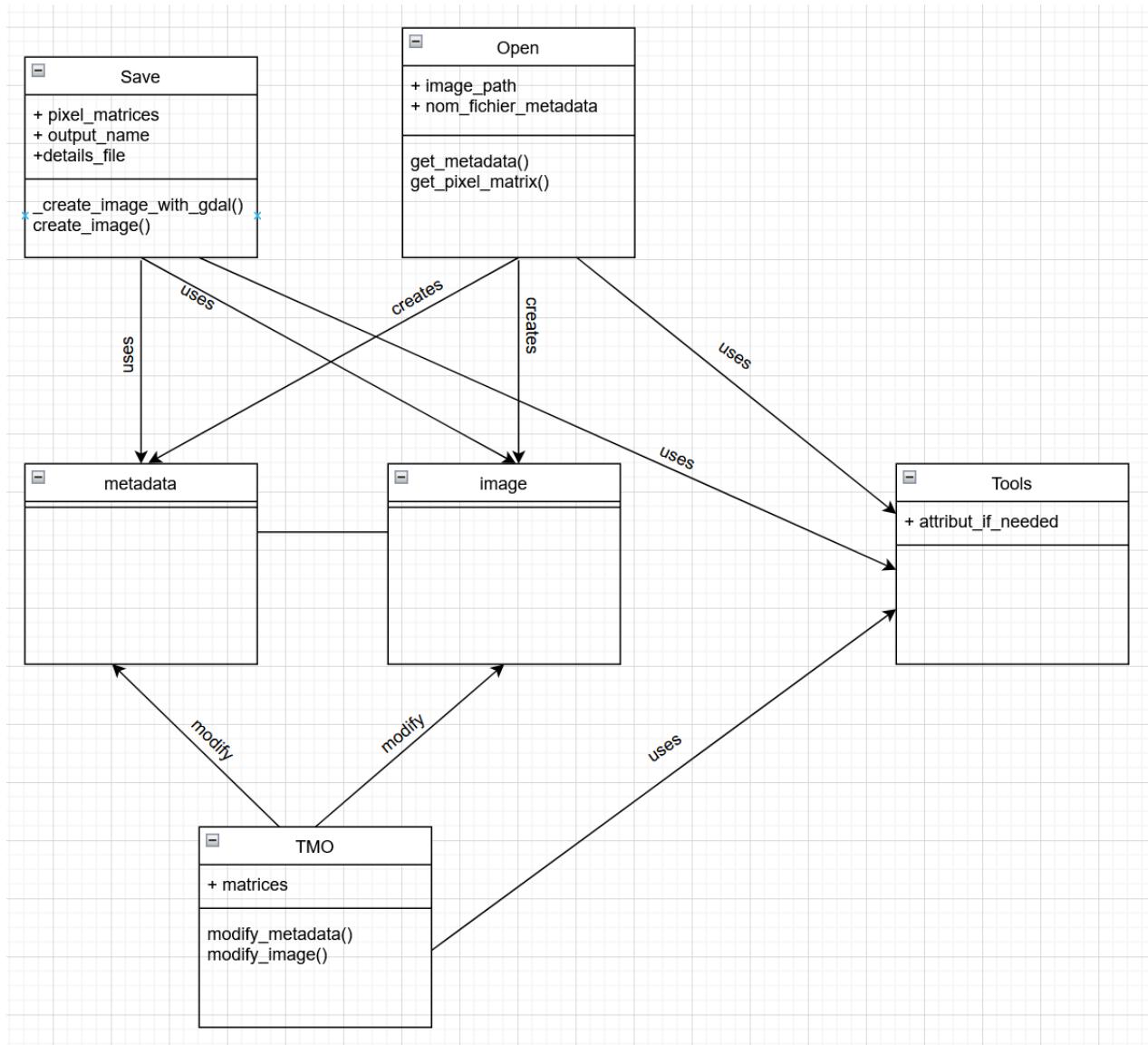


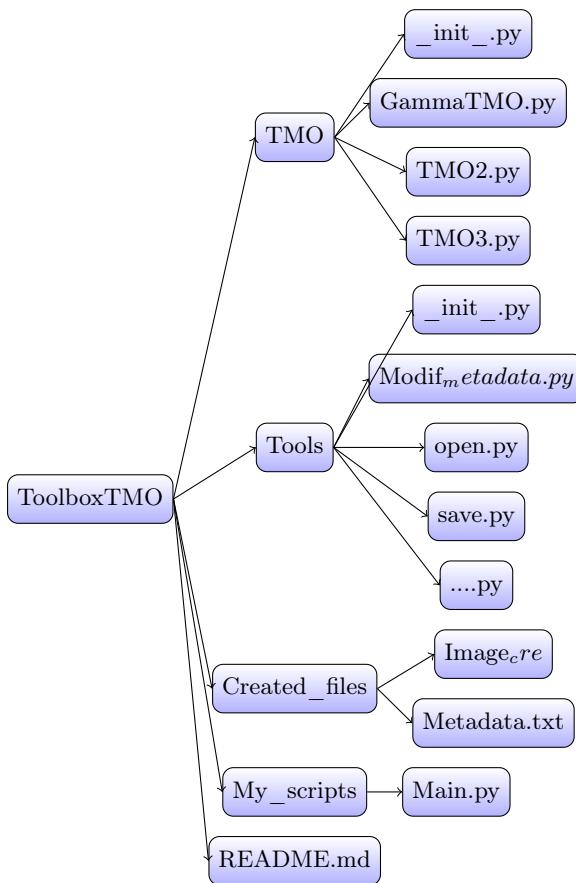
FIGURE 3 – Diagramme de Classes

4.2 Organisation du package

Par besoin de clarté, j'ai regroupé l'ensemble des TMO dans un dossier et placé tous les outils satellites dans un autre dossier 'tools'. Ainsi, je peux appeler les outils si mes chemins sont bien configurés.

Ayant rencontré de nombreuses difficultés dans l'import des classes dans d'autres fichiers, j'ai décidé de laisser de côté l'import de fonctions à l'aide du fichier `__init__.py` et d'utiliser des imports à l'aide de chemins relatifs. L'arborescence reste sensiblement la même que celle envisagée lors de l'analyse du projet.

Arborescence du dossier `TMO_toolbox`



4.3 Résultats obtenus

En appliquant la chaîne de traitement à une image géoréférencé, on récupère cette image modifiée avec les métadonnées complétés de l'opération effectuée.

Si on réutilise l'image et que l'on applique la fonction TMO inverse, celle-ci va récupérer les paramètres enregistrés dans les métadonnées pour effectuer l'opération inverse. On récupère deux images semblables.

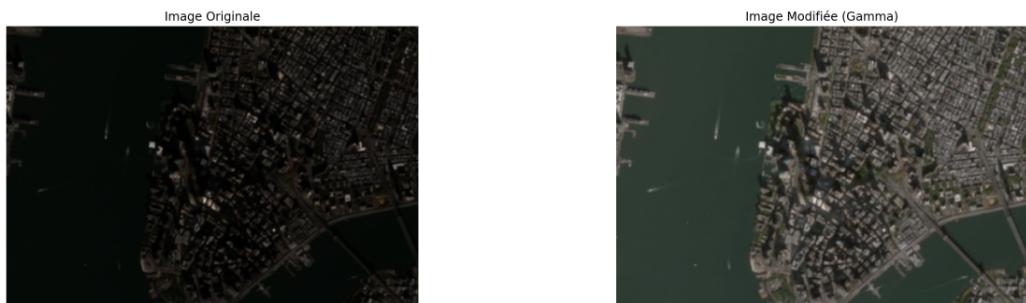


FIGURE 4 – Avant-après GammaTMO

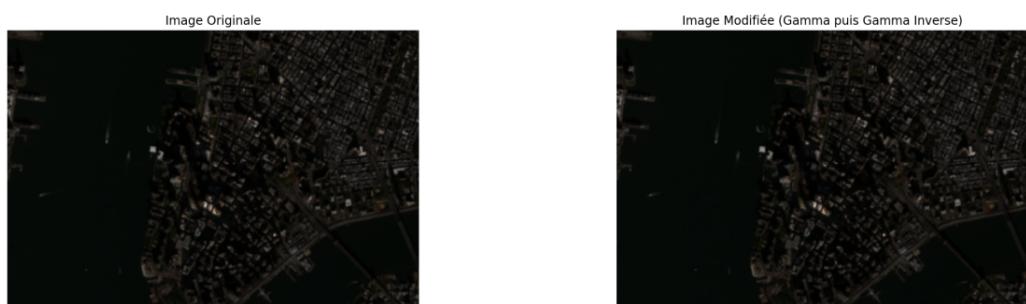


FIGURE 5 – Avant-Après GammaTMO et GammaInvTMO

```
Moyenne Image originale : 0.05960521474480629
Moyenne après Gammatmo : 0.25071895122528076
Moyenne après Gammatmo et inverse : 0.058766741305589676
Moyenne des différences entre image originale et finale : 0.0008384737302549183
```

FIGURE 6 – Différence de valeurs des pixels lorsque l'on applique le TMO et TMO inverse

La différence entre les valeurs de l'image avant et après TMO est de l'ordre du pourcent.

5 Gestions de projet

5.1 Les outils de suivi

Le point sur l'avancée du projet s'est fait principalement par réunion à l'ign puis à l'ENSG géomatique.

5.2 Les risques

Différents risques ont pu mettre à mal le projet. Je les avais pris en compte lors de la réalisation de la phase d'analyse. Dans mon cas, j'avais identifié comme menace la plus forte la difficulté d'implémentation. L'implémentation des TMO en eux-mêmes n'est pas aussi compliqué que prévu. Le risque lié aux bibliothèques à utiliser s'est révélé réel. Les librairies disponibles ne sont pas toutes adaptées aux données géoréférencées et elles ne permettent pas d'ouvrir tous les types de fichiers. En ajoutant à cela que certaines de ces fonctions ne sont utilisables que dans le cas d'images 8bits ou 16bits mais pas nécessairement les deux en même temps. Pour éviter de me perdre, je souhaitais donc commencer par décrire les limites de chaque bibliothèque et avancer étape par étape quitte à ne pas avoir de résultat présentable dans les premières séances. Le problème étant qu'il existe de nombreuses bibliothèques et de nombreux cas pour lesquels seulement une est utilisable. J'ai donc finalement codé en avançant à taton. En choisissant la fonction à utiliser une fois devant le problème clair.

Le manque de temps n'est pas tant une menace mais une réalité. Il y a de très nombreux TMO à implémenter. Je n'ai pas eu le temps d'avancer autant que prévu et je n'ai pu en implémenter qu'un. Ce n'est pas très grave puisque d'autres personnes pourront reprendre mon code et continuer.

Ensuite une perte de donnée aurait arriver si j'avais écrasé mon code en le poussant mal sur Github. J'avais donc gardé une archive sur disque dur.

Risques	Impact	Probabilité	Solution
Difficulté d'implémentation	fort	modéré	Avancer étape par étape
Manque de temps	faible	fort	Organiser clairement le projet pour permettre une continuation
Perte de données	fort	faible	Archivage sur disque dur et dépôt Github

FIGURE 7 – Tableau des risques

5.3 Le déroulement

Nous avons eu un total de 15 séances dédiées au projet informatique. La priorité était de pouvoir exécuter toute la chaîne de traitement avec au moins un TMO. Une fois que cette chaîne est opérationnelle, il suffit d'implémenter de nouveaux TMO, ceux-ci seront directement utilisables dans la chaîne. De plus, si l'architecture globale est claire, l'implémentation d'un TMO peut-être faite par un autre opérateur. J'ai réussi à atteindre ce premier objectif avec une chaîne entière.

Je pensais lors de l'analyse que créer les fonctions open et save serait une partie aisée. C'était sans prendre en compte la difficulté qu'apportait la création et la modification d'un fichier .txt pour les métadonnées mais aussi les nombreuses incompatibilité de certains fonctions avec certaines profondeurs d'images. L'implémentation des TMO semble finalement être la partie la plus aisée puisqu'elle consiste principalement à manipuler des matrices.

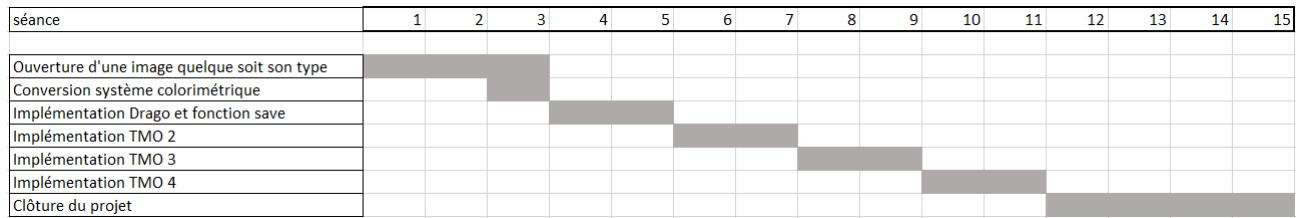


FIGURE 8 – Diagramme de Gantt prévu

Le déroulement du projet a été tout à fait différent. Mais Il colle avec les remarques qui m'avaient été faites lors de la soutenance d'analyse. Prévoir de nombreuses implémentations de TMO était ambitieux aux vues des autres fonctions à coder avant.

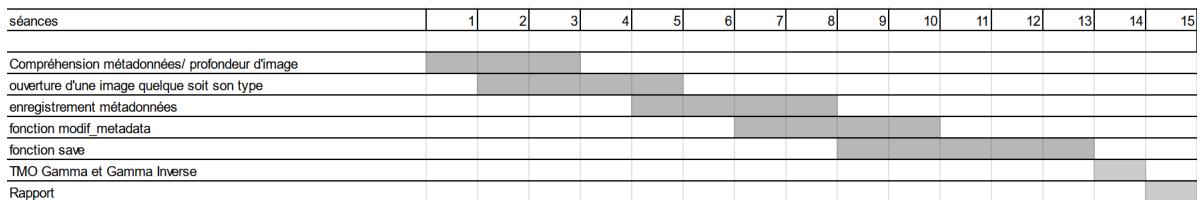


FIGURE 9 – Diagramme de Gantt de la réalisation du projet

5.4 Les perspectives

Les fonctions permettant le bon déroulement de la chaîne de traitements sont codées. La Toolbox est donc bien avancée mais pas finie. Il reste à implémenter les TMO nécessaires. Cela représente un certains travail à celui qui reprendra le code. Les métadonnées sont actuellement enregistré au format txt. Passer au format xml permettrait de construire quelque chose de plus solide en ayant un format d'écriture bien précis. Les fonctions permettent pour le moment de traiter des images 8 ou 16bits de profondeur. Il faudrait continuer et permettre l'utilisation d'images 24bits. Cela risque de nécessiter de retoucher à plusieurs fonctions comme open et save.

Références

- [1] F. Drago, K. Myszkowski, T. Annen and N. Chiba, *Human-in-the-loop development of spatially adaptive ground point filtering pipelines—An archaeological case study*, Article de recherche, 2003.

Table des figures

1	Image HDR [1]	1
2	diagramme d'utilisation	4
3	Diagramme de Classes	7
4	Avant-après GammaTMO	9
5	Avant-Après GammaTMO et GammaInvTMO	9
6	Différence de valeurs des pixels lorsque l'on applique le TMO et TMO inverse	9
7	Tableau des risques	10
8	Diagramme de Gantt prévu	11
9	Diagramme de Gantt de la réalisation du projet	11