Manuel de conception MAJA

Version 4.3.0

MAJA Team

déc. 16, 2020

Glossaire et Liste des paramètres AC & AD 1

CS Systèmes d'Information

Business Unit ESPACE
Département Payload Data & Applications
Image Processing

Software : **MAJA**			
SETG-CG-MAJA-010-CS			
Change	02	Date	29/11/2019
Issue	02	Date	05/05/2020
Distribution Code	E		
Ref. : CSSI/SPACE/PD&A/MAJA/DCG			

DOCUMENT DE CONCEPTION GLOBALE POUR LES CHAINES MAJA [DCG]

Rédigé par :	le : 05/05/2020
ESQUIS Benjamin CSSI/ESPACE/PDA/IP	
Validé par :	le : 05/05/2020
OLIVIE Francis CSSI/ESPACE/DSM	
Pour application :	le : 05/05/2020
ESQUIS Benjamin CSSI/ESPACE/PDA/IP	

Bordereau d'indexation

Confidentialité : DLP	Mots clés : Conception, MAJA, Sentinel2, VENµS, chaînes L2		
TITRE Du Document : Document de conception globale MAJA [DCG]			
AUTEUR(s) :	BRICIER Aurélien CSSI/ESPACE/PDA/IP ESQUIS Benjamin CSSI/ESPACE/PDA/IP		
RESUME : Ce document présente la conception définie pour le développement de l'outil MAJA.			
Documents Rattaches : Le document [SDM] complète ce document...	LOCALISATION : CSSI/SPACE/PD&A/MAJA		
Volume : 1	Nombre total de pages : N/A Dont pages préliminaires : 0 Nombre de pages suppl : 0	Doc composite : N	LANGUAGE : FR
GESTION DE CONF. : NG	CM RESP. :		
CAUSE D'EVOLUTION : Mise à jour pour MAJA v4.1.0			
CONTRAT : Marché ACSIS n°181112			

Diffusion interne

ESQUIS Benjamin	CSSI/ESPACE/PDA/IP
BROSSARD Julie	CSSI/ESPACE/PDA/IP
OLIVIE Francis	CSSI/ESPACE/DSM
RECULEAU SERGE	CSSI/ESPACE/PDA/PDGS

Diffusion externe

Name	Entity	Observations
BAILLARIN Simon	DNO/OT/IS	
HAGOLLE Olivier	DSO/SI/CB	
KETTIG Peter	DNO/OT/IS	
LARIF Marie-France	DNO/OT/PE	
SELLE Arnaud	DNO/OT/TA	
PACHOLCZYK Philippe	DNO/OT/TA	
PUJOL Mathilde	DNO/DA/AQ	

Modification

Ed.	Rév.	Date	Référence, Auteur(s), Causes d'évolution
02	02	05/05/2020	CSSI/ESPACE/PDA/IP/MAJA/DCG ESQUIS Benjamin CSSI/ESPACE/PDA/IP MAJ pour MAJA 4.2
02	01	10/03/2020	CSSI/ESPACE/PDA/IP/MAJA/DCG ESQUIS Benjamin CSSI/ESPACE/PDA/IP Refonte pour MAJA 4.1
02	00	29/11/2019	CSSI/ESPACE/PDA/IP/MAJA/DCG ESQUIS Benjamin CSSI/ESPACE/PDA/IP Refonte pour MAJA 4.0
01	03	18/04/2018	CSSI/ESPACE/PDA/IP/MAJA/DCG BRICIER Aurélien CSSI/ESPACE/PDA/IP Ajout des modifications pour MAJA 3.1
01	02	20/10/2017	CSSI/ESPACE/PDA/IP/MAJA/DCG BRICIER Aurélien CSSI/ESPACE/PDA/IP Ajout des modifications pour MAJA 3.0
01	01	04/05/2017	CSSI/ESPACE/PDA/IP/MAJA/DCG ESQUIS Benjamin CSSI/ESPACE/PDA/IP Mise à jour du document suite à la livraison de MAJA 2.0 Ajout description I3D
01	00	21/12/2016	CSSI/ESPACE/PDA/IP/MAJA/DCG FEUVRIER Thomas CSSI/ESPACE/PDA/IP Création du document à partir du document MACCS LAIG-CG-MAC-010-CS édition 02/04

CHAPITRE 1

Glossaire et Liste des paramètres AC & AD

OT	Organigramme Technique
PDL	Program Design Language

Liste des paramètres AC :

Liste des paramètres AD :

CHAPITRE 2

Généralités

2.1 Documents de référence

L'ensemble de la documentation de référence au projet MAJA est décrite dans le document [LD] référencé SETG-LD-MAJA-010-CS correspondant à la liste documentaire du projet et présentant de façon exhaustive et précise l'ensemble de la documentation de référence du projet.

2.2 Documents applicables

L'ensemble de la documentation applicable au projet MAJA est décrite dans le document [LD] référencé SETG-LD-MAJA-010-CS correspondant à la liste documentaire du projet et présentant de façon exhaustive et précise l'ensemble de la documentation applicable au projet.

CHAPITRE 3

introduction

3.1 objet du document

L'objet de ce document est de présenter la conception globale de MAJA. MAJA s'appuie sur la chaîne L2 développées. Pour éviter la redite, le document de conception globale de la chaine chaînes L2 [DA100] s'applique à ce document. Seuls les points de conception spécifiques à MAJA sont alors présentés et justifiés dans ce document. Le document [SDM] (Software Developer Manual) vient compléter la description faite dans ce document. En outre, les aspects purement informatique tels que l'organisation de l'arborescence, la constitution des plugins, etc.... sont décrits dans le [SDM].

3.2 structure du document

Ce document s'articule autour des grands chapitres suivants :

- Le chapitre *Rappel des principales exigences et contraintes* présente les principales contraintes et exigences de MAJA,
- Le chapitre *Présentation générale de MAJA* décrit dans sa globalité l'architecture et les interfaces retenues pour MAJA,
- Le chapitre *Les algorithmes* décrit les points de conception de MAJA pour la partie purement algorithmique de MAJA,
- Le chapitre *Les plugins « capteur »* contient les points de conception de MAJA pour la partie purement plugin.

CHAPITRE 4

Rappel des principales exigences et contraintes

4.1 Exigences et contraintes générales

L'outil MAJA (MACCS ATCOR Joint Algorithms) combine plusieurs traitements :

- L'estimation de masques de :
  - Couverture nuageuse et ombres des nuages
  - Eau
  - Neige
  - Ombres de relief
- L'estimation du contenu en vapeur d'eau
- L'estimation de l'épaisseur optique des aérosols
- La correction atmosphérique en utilisant des données exogènes (METEO, MNT, . . .) et l'épaisseur optique des aérosols. La correction atmosphérique inclut aussi une correction des effets d'environnement.
- La correction des variations d'éclairement dues aux pentes en utilisant l'information fournie avec le MNT et en considérant une fonction de distribution de la réflectance bidirectionnelle.

A partir de la version 3.0.0, l'architecture de MAJA a été modifiée afin de prendre en compte les exigences de généricité. L'outil est par conception **multi capteurs**, et chaque capteur est désormais géré dans un « plugin ». Dans la version 3.0.0, les plugins VENUS et SENTINEL2 sont implémentés ; les plugins FORMOSAT et LANDSAT seront implémentés dans la prochaine version.

En intégrant les produits Sentinel-2 à la chaîne MAJA, elle doit maintenant gérer des **résolutions multiples** dans un même produit de niveau 1.

La version 4.0.0 à marqué un tournant dans MAJA. En effet l'ensemble du logiciel à été remanié pour permettre une plus grande souplesse quand aux futurs développements. L'enchainement des algorithms ainsi que la lecture des produits se font dorénavant en python. Les algorithms quand à eux sont implémentés sous forme d'applications OTB indépendantes. Cela permet alors de pouvoir tester chaque algorithm indépendament de MAJA, de les modifiés simplement ou encore de les utilisés dans un contexte différent. Cela permet également à d'autres contributeurs de pouvoir fournir une application à integrer dans MAJA.

A noté également qu'a partir de la version 4.0.0 Maja n'implémente plus certains plugins obsolètes.

L'outil MAJA doit pouvoir être utilisé sous deux formes :

— Fonctionnement intégré dans une structure d'accueil (VIP dans le projet Venμs) permettant un fonctionnement en mode automatique. Dans ce cas, le lancement des traitements utilisera le mode d'utilisation défini dans le segment sol de Venμs.
— Fonctionnement avec une interface en mode ligne (dit aussi « stand alone ») qui pourra être utilisée dans un environnement Sentinel-2 (SL2P ou IQP) ou dans le futur Centre de Traitement National multi-missions.

L'applicatif MAJA s'appuie sur différents types de modules :

En c++ :
— Modules (Bibliothèques) métiers réalisant les différents traitements déjà décrits plus haut, sous forme d'applications indépendantes

En python :
— Modules (Bibliothèques) de gestion et d'accès aux images/masques etc...
— Modules d'accès aux fichiers de données (XML, de type GIPPs...).

Répond à : Exigence 10 dans : [DA01] Conformité : C

4.2 Exigences d'interfaces

MAJA 4.1.0 peut traiter en entrée des produits de niveau 1 :
— Venus (format MUSCATE)
— Venus (format NATIF)
— Landsat 8 (format MUSCATE)
— Landsat 8 (format NATIF)
— Sentinel2 (format MUSCATE)
— Sentinel2 (format natif)

Pour ce faire, les couches génériques (dites « Factory ») de lecture des produits et informations connexes (métadonnées) sont développées pour les cas suivants :
— Venus (format MUSCATE)
— Venus (format Natif)
— Sentinel-2A 1-C (format MUSCATE)
— Sentinel-2B 1-C (format MUSCATE)
— Landsat 8 (format MUSCATE)
— Landsat 8 (format Natif)
— Sentinel-2A 1-C (format natif)
— Sentinel-2B 1-C (format natif)

La trans-écriture de Natif vers Muscate est nativement possible dans Maja et est activée par défaut sauf si l'option de choix du format de sortie est utilisée.

A noter que certains plugins ne permettent pas l'écriture vers le format Natif mais uniquement la lecture ainsi le produit de sortie sera obligatoirement au format muscate :
— Venus Natif
— Landsat 8 Natif

Remarque : Pour Sentinel 2, un produit niveau 1-C couvre une « tuile » de 100 km x 100 km (selon la définition de « tuile/tile » indiquée dans le [DA05]).

Répond à : Exigence 90 dans : [DA01] Conformité : C

Répond à : Exigence 20 dans : [DA01] Conformité : C

Répond à : Exigence 400 dans : [DA01] Conformité : C

Répond à : Exigence 410 dans : [DA01] Conformité : C

Répond à : Exigence 470 dans : [DA01] Conformité : C

Répond à : Exigence 580 dans : [DA01] Conformité : C

Répond à : Exigence 590 dans : [DA01] Conformité : C

Répond à : Exigence 620 dans : [DA01] Conformité : C

Répond à : Exigence 630 dans : [DA01] Conformité : C
Répond à : Exigence 640 dans : [DA01] Conformité : C
Répond à : Exigence 40 dans : [DA01] Conformité : C
Répond à : Exigence 420 dans : [DA01] Conformité : C
Répond à : Exigence 430 dans : [DA01] Conformité : C

4.3 Exigences des traitements

Les traitements algorithmiques MACCS sont modifiés afin de prendre en compte les produits Sentinel-2 (Level 1-C). Ainsi, les évolutions algorithmiques décrites dans cette section prennent en compte les évolutions décrites dans le [DA02].

Répond à : Exigence 30 dans : [DA01] Conformité : C

4.4 Exigences de performances

Répond à : Exigence 160 dans : [DA01] Conformité : C

Ne pas dégrader les performances initiales de MACCS (performances initialement établies sur les chaînes L2/L3 VENμS) conditionne la conception et les développements à suivre.

Concrètement, dès la phase de développement, on vérifie que les temps d'exécution des TVs de référence VENμS ne sont pas dégradés dans MACCS. Ainsi, tout comme les TUs et les TVAs, les tests de performance seront lancés toutes les nuits, avec un contrôle régulier du temps d'exécution des tests VENμS par les membres de l'équipe.

4.5 Exigences de généricité

Répond à : Exigence 300 dans : [DA01] Conformité : C
Répond à : Exigence 330 dans : [DA01] Conformité : C

La chaîne MAJA est un seul et même logiciel configurable quel que soit le nombre de capteurs et les nombre de centres utilisateurs. La chaine de traitement MAJA peut être générée pour traiter les produits de tous les capteurs, d'un sous-ensemble de capteurs ou d'un capteur unique. En effet, des options de compilation permettent d'activer les capteurs souhaités. Ainsi, il est par exemple possible de générer le paquet binaire de MAJA pour le seul capteur VENUS.

Répond à : Exigence 310 dans : [DA01] Conformité : C
Répond à : Exigence 320 dans : [DA01] Conformité : C

Toutes les spécificités liées à un capteur sont prises en charge par le Plugin (code source et schémas d'interface). Ainsi, la chaîne de traitement MAJA peut traiter les produits d'un nouveau capteur sans que le traitement des produits des autres capteurs ne soit impacté, et la modification des interfaces entre la chaîne MAJA et un capteur n'impacte pas les interfaces des autres capteurs avec la chaine MAJA.

CHAPITRE 5

Présentation générale de MAJA

5.1 Chaînes opérationnelles

Le principe des chaînes opérationnelles est exactement le même que celles des chaînes VENµS L2/L3. Elles sont simplement ajustées pour être conforme à MAJA.

5.1.1 Identification des chaines

Les cinq chaînes scientifiques identifiées sont :
- MAJA_L2_INIT_CHAIN,
- MAJA_L2_NOMINAL_CHAIN,
- MAJA_L2_BACKWARD_CHAIN,

5.1.2 Cas d'utilisation de MAJA

Lancée par l'utilisateur MAJA ou par un centre utilisateur, avec en ligne de commande (cf. [MU]) les paramètres (sous forme d'options), l'application **maja** exécute au choix l'un des traitements listés section précédente *Identification des chaines*.

On note que l'application **maja** n'a pas besoin de TaskTable pour fonctionner.

Ainsi, c'est le centre opérationnel qui va enchaîner successivement les traitements (dans ce cas un seul traitement), définis dans la TaskTable **« VIP »**. Si le CNES désire installer MAJA au VIP, il sera nécessaire de définir de nouvelles TaskTables spécifiques à MAJA.

5.1.3 L'application "maja"

L'application **maja** est modifiée, elle enchaîne désormais les traitements en lançant des applications OTB des algorithms. La lecture et l'écriture des produits de faits en python, de même que l'enchainement des algorithms.

MAJA est **multi-satellite**. En effet, la chaîne peut traiter des produits de niveau 1 provenant de la même mission, mais de capteurs différents. Par exemple, pour le cas Sentinel2, la chaîne ingère indifféremment des produits S2A, S2B et S2C.

Dans cette version de MAJA, il est possible de fournir en sortie des produits au format MUSCATE à partir de produits au format Natif.

Note importante : la description des plugins (et la procédure d'implémentation) est décrite en détail dans le Software Developer Manual [SDM].

5.2 Interfaces

5.2.1 Présentation générale

Le principe retenu sur le format des données produites en sortie est le suivant : chaque produit L2 doit être conforme à son format « capteur » d'origine. La constitution (format, nombre de fichiers, etc...) d'un produit L2 est donc dépendante du satellite. En d'autres termes, un produit VENµS de niveau 2 doit être conforme à l'ICD Venµs, un produit Sentinel-2 de niveau 2 doit être conforme à l'ICD S2, etc...

Dans les faits, les schémas des produits correspondant à chacun des capteurs sont gérés par capteur, dans le **Plugin** correspondant. Ainsi chaque capteur possède ses propres schémas de produits (L1, L2 et L3).

De manière générale, tous les schémas des produits, des fichiers GIPPs et des fichiers de configuration liés au capteur sont gérés dans le plugin dudit capteur.

Se reporter au chapitre 3 du [SDM] pour une présentation détaillée des interfaces externes/internes de MAJA.

5.2.2 Interface des produits images

Répond à : Exigence 70 dans : [DA01] Conformité : PC
Répond à : Exigence 80 dans : [DA01] Conformité : PC

5.2.2.1 Produits de NIVEAU 1

Le format des produits de niveau 1 VENµS est inchangé.

5.2.2.1.1 Produits L1 Sentinel-2 (niveau 1-C) (S2A et S2B)

Les méta-données

La chaîne contrôle la cohérence et l'intégrité des données Sentinel-2 de niveau 1-C par l'application des schémas de l' « *ICD-Sentinel-2* » [PSD] pour les produits issus du PDGS.

Les informations nécessaires aux traitements algorithmiques sont présentes dans les fichiers d'entête du produit Sentinel-2.

Une des spécificités des produits Sentinel-2 par rapport au produit Venµs est la fourniture des masques associés à un produit de niveau 1 sous forme vecteur (format GML) dans le header du produit (masques des pixels saturés, aberrants, no_data).

La valeur du **No_Data** est renseignée dans le nœud Product_Characteristics/Product_Image_Characteristics/Special_Values/, il y a deux valeurs définies :

```
SPECIAL_VALUE_TEXT : NODATA
SPECIAL_VALUE_INDEX : 0
```

Il existe également un masque des No_Data au format vecteur dans les fichiers d'entête. Ce masque est utilisé pour générer le masque des No_Data au format image utilisé dans la chaîne.

Les informations sur les **bandes spectrales** sont définies dans le nœud : Product_Characteristics/Spectral_Information_List :

```
<Spectral_Information band_id="0" physical_band="B1">
     <RESOLUTION>60</RESOLUTION>
     <Wavelenght>
          <MIN unit="nm">414</MIN>
          <MAX unit="nm">472</MAX>
          <CENTRAL unit="nm">443.0</CENTRAL>
     </Wavelenght>
     <Spectral_Response>
          <STEP unit="nm">1.0</STEP>
          <VALUES>0.00000726 0.0000082 0.00000245 ...</VALUES>
     </Spectral_Response>
</Spectral_Information>
```

Les informations sur les tuiles sont contenues dans le nœud Data_Strip/Tiles_List/Tile, avec un « Id » unique par tuile. La chaîne MAJA ne traite qu'une tuile. Il est donc nécessaire de récupérer l'Id de la tuile traitée pour aller chercher les informations dans le bon nœud « Tile » du header principal.

Les données SOL et VIE sont décrites dans le Header du produit 1-C. Par exemple, une liste de nœuds nommés « Viewing_Incidence_Angles_Grids » définissent les angles pour chaque bande et pour chaque zone. Ces informations sont extraites du header pour reconstituer l'image des angles de prises de vues et en déduire également le masque des zones.

Le header principal fournit donc :

- les angles solaires et les angles de prises de vue. Ils sont définis respectivement dans les nœuds « Sun_Angles_Grid » et « Viewing_Incidence_Angles_Grids ».
- le chemin pointant sur les fichiers des masques (format GML).

Dans les fichiers de masque (dans le nœud Mask_List/MASK_FILENAME), il y a pour chaque bande :

- Les pixels aberrants,
- Les pixels saturés.

  Ces données sont à lire afin d'extraire ces informations et en faire des masques (rasterisation).

Images

La donnée image est au format JPEG2000, avec 13 fichiers, un par bande spectrale.

Les fichiers sont listés dans le header xml, dans le nœud Product_Characteristics/Product_Image_Characteristics/Data_Access/DATA_FI
Cependant, il faut s'assurer que les noms des fichiers lus correspondent bien à la tuile traitée.

La lecture des images JPEG2000 est réalisée avec l'OTB.

5.2.2.1.2 Les produits au format MUSCATE

Les produits de niveau 1 Landsat 5/7/8 et Sentinel2 (S2A et S2B) au format MUSCATE sont lus en respectant les interfaces définies dans le [PSC].

5.2.2.2 Produits de NIVEAU 2

Les produits de niveau 2 VENμS sont écrits en respectant les interfaces [DA06] et [DA07].

La chaîne MAJA génère des produits de niveau 2 au format « EarthExplorer » pour les capteurs Formosat, Sentinel-2 GPP, Sentinel2 natif PDGS (S2A et S2B), Landsat (L5 et L7), Landsat 8 et Spot4. Ces produits sont conformes à un format qui est défini dans une nouvelle interface appellée « **ICD-MAJA** ». Elle assure la conformité de tous les produits générés par MAJA dans le format défini pour chaque satellite, tout en s'assurant qu'un produit VENμS est bien conforme à l'ICD Venμs.

Les produits de niveau 2 Sentinel2 GPP (S2A et S2B) sont écrits en respectant les interfaces définies dans le Plugin SENTINEL2_GPP (création des schémas de ces produits de niveau 2 dans un ICD Sentinel2_GPP).

Les produits de niveau 2 Sentinel2 PDGS (S2A et S2B) sont écrits en respectant les interfaces définies dans le Plugin SENTINEL2 (création des schémas de ces produits dans un ICD Sentinel2).

Les produits de niveau 2 Landsat 8 natif sont écrits en respectant les interfaces définies dans le Plugin LANDSAT8 (création des schémas de ces produits dans un ICD LANDSAT8).

Une description détaillée (sous forme de tableaux Excel) de la constitution de ces produits est présentée dans le [MU].

Répond à : Exigence 50 dans : [DA01] Conformité : C
Répond à : Exigence 60 dans : [DA01] Conformité : C
Répond à : Exigence 440 dans : [DA01] Conformité : C
Répond à : Exigence 450 dans : [DA01] Conformité : C
Répond à : Exigence 600 dans : [DA01] Conformité : C
Répond à : Exigence 610 dans : [DA01] Conformité : C
Répond à : Exigence 650 dans : [DA01] Conformité : C

Le produit de niveau 2A généré à partir du produit Sentinel-2 1-C est écrit dans un format inspiré du format VENμS comme décrit dans le [DA06].

Le produit est constitué d'un fichier d'entête .HDR contenant toutes les informations générales du produit image (comme un produit VENUS) et d'un répertoire .DBL.DIR contenant les fichiers constitutifs du produit image.

Le produit de niveau 2, généré à partir de produits Landsat 5 et 7, est écrit dans un format inspiré du format VENμS comme décrit dans le [DA06]. Il ressemble aux produits Venμs et Formosat de niveau 2 actuellement générés par la chaîne L2 VENμS.

Les données lues et écrites dans la partie privée, ne sont pas impactées par la multi-résolution de Sentinel-2, elles sont écrites à la résolution réduite des produits L2 (L2Coarse).

Produits au format MUSCATE :

Les produits de niveau 2 Landsat 5/7/8 et Sentinel2 (S2A et S2B) au format MUSCATE sont écrits en respectant les interfaces définies dans le [PSC].

5.2.3 Interface des fichiers de configuration

Les fichiers de configuration sont les suivants :

- MAJA**Admin**ConfigSystem.xml : les paramètres de configuration destinés au fonctionnement de la chaîne et ne devant en aucun cas être modifiés par l'utilisateur,
- MAJA**User**ConfigSystem.xml : les paramètres de configuration utilisateurs communs à la chaîne, ne dépendant pas du capteur,
- MAJA**User**Config_<CAPTEUR>.xml : les paramètres de configuration utilisateurs dépendant du capteur.
- MAJA**Admin**Config_<CAPTEUR>.xml : les paramètres de configuration administrateurs dépendant du capteur.

Les paramètres de configuration dépendant du capteur et destinés au fonctionnement de la chaîne sont pris en charge par le plugin et implémentés dans le code source. Leur schéma est défini dans les schémas au niveau du **Plugin**.

La solution technique implémentée permet de générer un produit de niveau 2 qui sera identique quelles que soient les conditions de « génération », c'est-à-dire qu'il ait été généré :

- en mode INIT ou
- en mode BACKWARD ou NOMINAL avec un produit de niveau 2 en entrée issu du même capteur ou non.

Ainsi dans le produit de niveau 2 les produits de réflectances de la partie PRIVATE contiendront un nombre identique de bandes, quelque sois le capteur traité (à une condition près, décrit ci-dessous).

Note : le fichier de configuration MAJA**Admin**Config_<CAPTEUR>.xml contient la liste des longueurs d'ondes théoriques pour chaque bande du capteur.

La gestion des résolutions est revue. Elle est désormais gérée dans le code.

Note : les modifications apportées au fichier de configuration n'impacte aucunement l'interface avec le VIP.

5.2.4 Interface des GIPPs

La conception a permis de dire qu'aucun nouveau GIPP n'est nécessaire pour les besoins de MAJA ; seuls les GIPPs actuels sont revus.

Par exemple, le GIPP principal GIP_L2COMM est mis à jour. Le nœud contenant la définition des bandes et leurs indices informatiques est déplacé dans le fichier de configuration (ou dans les factories de lecture/écriture des produits). Les nouveaux paramètres tels que ceux liés au paramétrage du SnowMasking sont créés.

Pour connaître en détail les éléments supprimés/modifiés/ajoutés des GIPPs, il convient de se reporter aux GIPPs mis à jour et livrés en fin de phase de conception. Ces GIPPs sont utilisés par le CNES pour définir les contextes de validation de la maquette.

Remarque : La phase de développement a permis d'identifier la création de paramètres dont les valeurs dépendent du satellite. En effet, pour l'algorithme d'AOT_Estimation, il est nécessaire de définir, dans le cas par exemple de Sentinel2, des valeurs pour les deux paramètres suivant « MS_Slope » et « MS_Yintercept » différentes pour le satellite S2A et pour le satellite S2B. Ces valeurs restent définies dans le fichier GIP L2COMM ; en conséquence, il est décidé qu'il y a en entrée de MAJA autant de fichier GIP_L2COMM que de satellite traité. Ainsi par exemple pour le cas Sentinel2, les deux fichiers S2A et S2B seront exigés en entrée de MAJA.

Le GIPP L2COMM est modifié. Il contient désormais **seulement** les paramètres des algorithmes activés pour le **capteur** concerné. Ainsi, pour le cas VENUS, les paramètres du SnowMasking n'existent plus. En conséquence, tous les schémas des fichiers GIPPs sont désormais définis dans l'ICD du **plugin** concerné, assurant ainsi l'indépendance des interfaces des GIPPs d'un capteur à l'autre.

Pour la version MACCS 4.0, le GIP_L2COMM a une nouvelle fois été mis à jour. Des paramètres communs ont été déplacés tel que le paramètre Slope_Min_Cos_I désormais utilisé dans les algorithmes Snow Masking et Slope Correction. De nouveaux paramètres ont également été ajoutés suite aux évolutions algorithmiques apportées dans cette version. La liste des algorithmes modifiés et les évolutions sont décrites dans *Description des evolutions algorithmiques de la chaine*.

5.3 Arborescence du code source

La mise en œuvre du MAJA générique a eu pour conséquence une modification importante de l'arborescence du projet.

Cmake est utilisé pour compiler le projet MAJA et ses dépendances.

Les répertoires « Testing », "**Checking**", "**CMakeConfig**", "**ddc**" et "**Doxygen**", non décrits ici, sont utilisés pour gérer le projet.

Le répertoire « make » contient les fichiers permettant de définir les options de compilations pour les modes « release » et « debug ») en fonction du compilateur sélectionné (ex : gcc ou icc).

Le répertoire « **Utilities »** contient les outils externes intégré au projet. On trouve par exemple le répertoire « I3D » contient des algorithmes pour calculer une direction à partir d'angles de visée et des angles solaires en utilisant un MNT.

Le répertoire « **orchestrator»** contient le code python de l'orchestrateur. Il est en charge de lire/écrire les produits dans les differents formats de capteur. Il enchaine les algorithmes en fonction du mode choisis (INIT, NOMINAL et BACKWARD) Le répertoire « **orchestrator»** est organisé comme suit :

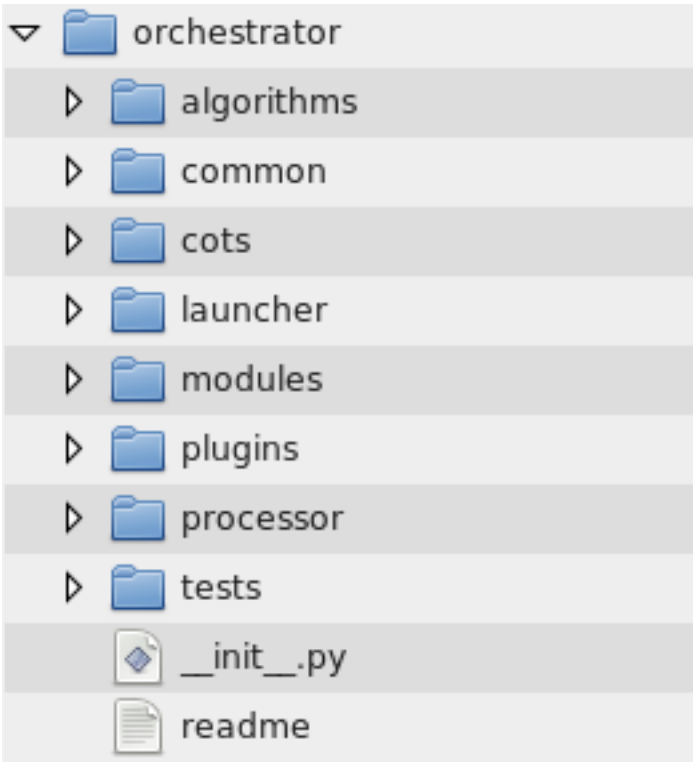


Figure 6 : Vue générale du répertoire « orchestrator » de MAJA

Ce répertoire est constitué des sous répertoires suivants :

- Le répertoire « **algorithms »** contient le code source des algorithmes implémentés en python dans MAJA. Ce répertoire est détaillé dans le chapitre 5 « Les algorithmes ».
- Le répertoire « **common »** contient les fonctions communes transverses comme les outils de lecture/ecriture des xml … Il est également constitués des sous-répertoires suivants :
- Le répertoire « conf » contient des fonctionnalités de configuration notamment pour lire les fichiers de configuration des plugins algorithmes
- Le répertoire « dem » contient le code source relatif à la lecture des dem
- Le répertoire « earth_explorer » contient les éléments commun relatifs au format earth explorer.

- Le répertoire « interfaces » contient les outils de lecture/ecriture des fichiers interfaces avec les applications. En effet certaines interfaces des applications algorithmes sont trop complexes pour être mise en paramètres et dons passe sous forme de xml.
- Le répertoire « logger » contient le code relatif à la configuration du logger de l'ordonanceur.
- Le répertoire « muscate » contient les éléments commun relatifs au format muscate.

- Le répertoire « **cots »** contient les wrappers de cots externe comme l'otb ou gdal
- Le répertoire « **launcher »** contient le code du lanceur d'application permettant notamment de lire les paramètres de l'application
- Le répertoire « **modules »** contient la définition de l'interface d'un module. Ceci est une interface générique de module de traitement. Une factory permet alors de fournir une instance de l'algorithme. Par exemple getinstanceof(« CirrusCorrection ») renvois un module de correction des cirrus.
- Le répertoire « **plugins »** contient le code python permettant de gérer la lecture et l'ecriture des differents formats de produits capteurs.
- Le répertoire « **processor »** contient le code python des differentes chaines INIT/NOMINAL/BACKWARD.
- Le répertoire « **tests »** contient l'ensembles des tests du code pythonde l'orchestrateur.

Le répertoire « **Packaging »** contient les procédure cmake permettant de generer le paquet binaire « .run » de maja.

Le répértoire « **Superbuild »** contient les fichiers CMake permettant de construire Maja à l'aide des SuperbuildArchives qui contiennent l'ensemble des dépendances, afin de ne pas avoir à installer les dépendances sur la machine. De plus cela permet de créer une version de Maja incluant l'ensemble de ses dépendances.

Le répertoire « **Plugins »** contient l'ensemble des ressources (xsd,GIPPs et templates) des differents capteurs. Ce répertoire est décrit dans le [SDM].

Le répertoire « **Code »** est organisé comme suit :

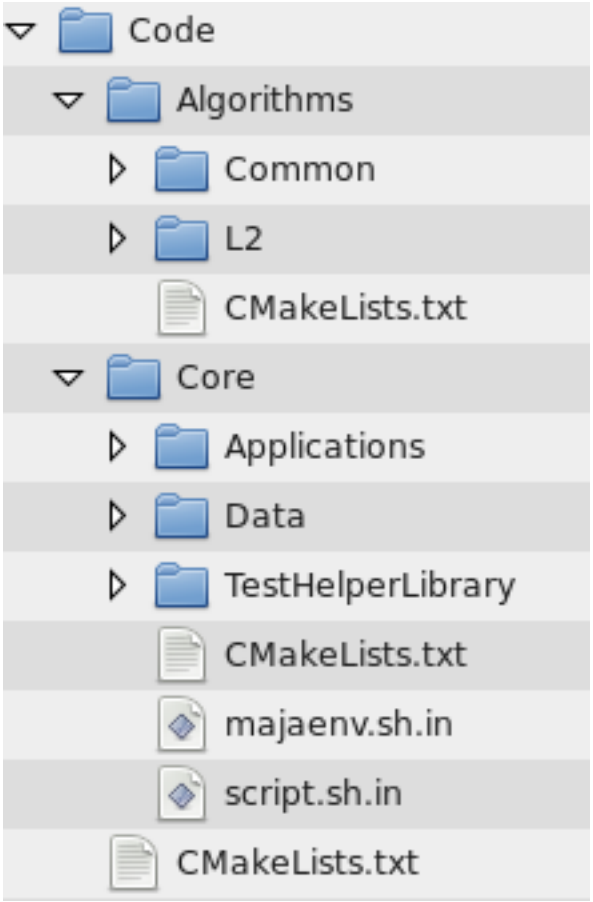


Figure 6 : Vue générale du répertoire « Code » de MAJA

Ce répertoire contient tout le code source de MAJA qui ne dépend pas des produits gérés et donc des plugins. Ce répertoire est constitué des sous répertoires suivants :

- Le répertoire « **Algorithms »** contient le code source des algorithmes implémentés dans MAJA. Ce répertoire est détaillé dans le chapitre 5 « Les algorithmes ».
- Le répertoire « **Core »** est constitués des sous-répertoires suivants :
- Le répertoire « Data » contient des fonctionnalités de bases et rendent des services aux couches de plus haut niveau telles que les algorithmes
- Le répertoire « Application » contient le code source des exécutables C++ de MAJA (un ou plusieurs exécutables pour chaque algorithme, ainsi que des applications communes transverses)
- Le répertoire « Scripts » contient l'ensemble des scripts du projet (.sh and .py). Il contient par exemple les scripts principaux « **maja** » (fichier bash et python).

Attention : Ce répertoire « Code » n'est pas impacté par l'ajout d'un nouveau Plugin, du moment où celui-ci n'implique pas l'ajout d'un nouvel algorithme ou la modification d'un existant.

5.4 Arborescence de déploiement

L'arborescence de déploiement est décrite en détail dans le [MU].

5.5 Architecture statique

Du fait de l'utilisation de deux langages au sein de MAJA deux architecture se cotoient. D'un coté les algorithmes, codés en C++, et de l'autre l'orchestrateur, codé en Python.

5.6 Architecture de la partie c++

Les différentes couches C++ de MAJA se présentent comme ceci (du bas vers le haut) :

- La couche « **Data** » contient l'ensemble des fonctionnalités de bases et offrent des services au couches supérieures. Par exemple, elle contient des classes pour faciliter la manipulation de fichiers XML (lecture, écriture, ...), les classes de base pour la gestion des loggers, des classes d'exceptions pour gérer les erreurs, les classes de bases pour lire les LUTs ou les fichiers d'interfaces.
- La couche « **Algorithms** » contient l'ensemble des classes implémentant les algorithmes de MAJA, celles-ci sont alors encapsulées dans des applications OTB afin d'être appelées par l'orchestrateur python.
- La couche « **Script** » contient le script principal **maja** ainsi que des scripts de haut niveau utilisés pour lancer MAJA

5.7 Architecture de la partie Python

Les différentes couches Python de MAJA se présentent comme ceci (du bas vers le haut) :

- La couche « **Data** » contient l'ensemble des fonctionnalités de bases et offrent des services au couches supérieures. Par exemple, elle contient des classes pour faciliter la manipulation de fichiers XML (lecture, écriture, ...), les classes de base pour la gestion des loggers, des classes d'exceptions pour gérer les erreurs, les classes de bases pour lire les fichiers d'interfaces et de configuration.
- La couche « **Plugins** » contient l'ensemble des classes implémentant les plugins de MAJA. Les plugins sont implémentés sous forme de factory. Pour chaque produit que l'on souhaite lire on demande à la factory qui serait capable de le lire. Chaque plugin intègre :

  - Un lecteur d'informations sur le produits L1
  - Un lecteur de produit L1
  - Un lecteur de produit L2
  - Un writer de produit L2 (métadonnées et images)
- La couche « **Chain** » contient l'enchainement permettant de passer d'un produit L1/L2 à un produit de sortie L2. Les chaines disponible sont décrite dans le [MU]

On note que l'on s'appuie sur l'OTB pour lire les fichiers JPEG2000 du produit Sentinel-2. Les masques des produits Sentinel-2 au format GML sont lus avec GDAL.

Les évolutions nécessaires pour implémenter MAJA sont présentées dans les sections suivantes.

5.8 Les composants logiciels

5.8.1 Composants Data Manager

5.8.1.1 Présentation

De manière générale, toutes les procédures de lecture et d'écriture sur les produits de niveaux 1, 2 sont prises en charge et implémentées dans les plugins. L'architecture est décrite en détail dans le document [SDM].

5.8.1.1.1 Phase d' « Initialisation » et management principal de l'application

L'initialisation des paramètres généraux de l'application ainsi que la lecture des fichiers de configuration système sont effectués en python par la classe AppHandler du launcher. Celle-ci récupère les paramètres et initialize un contexte d'éxécution.

5.8.1.1.2 Lecture des angles de prises de vue :

Dans la chaîne MAJA, aucune modification n'est apportée aux calculs des angles de prises de vue réalisés dans la chaîne VENµS L2 pour les capteurs Venµs et Formosat :

- pour Venµs, des grilles contenant les décalages des pixels en ligne/colonne (X,Y) sont fournies avec les produits de niveau 1. Toutefois, la variation des angles dans le champ n'est prise en compte que pour la détection des ombres et des faces cachées. Les autres algorithmes utilisent les valeurs des angles au centre de l'image.
- pour Formosat, ces grilles n'existent pas et sont générées par la chaîne à partir des valeurs d'angles solaires et de visée constants sur toute l'image.

Les grilles d'angles des capteurs Sentinel-2 et Landsat sont également générées au niveau de cette Factory.

Comme pour Formosat, Landsat, Landsat8 et Spot4 ne disposent pas de ces grilles en entrée. Des images constantes sont donc générées à partir des valeurs des angles solaires et de visée au centre de l'image.

En revanche, Sentinel-2 est un capteur à champ large pour lequel la variation des angles dans le champ ne peut plus être négligée. Il est donc nécessaire de disposer de la valeur de ces angles par pixel. Pour se ramener au format utilisé dans la chaîne VENµS, des grilles d'angles (image raster) doivent être produites. Ces grilles d'angles de prises de vue sont générées à partir des images vecteurs contenues dans le header principal d'un produit Sentinel-2.

Pour les angles solaires, ces images vecteurs sont simplement converties en images raster et les valeurs d'angles sont ramenées en position (X,Y) dans l'image. Les grilles ainsi obtenues sont ensuite ré échantillonnées à la résolution réduite des produits L2.

Pour les angles de visée, des images vecteurs sont fournies pour chaque bande spectrale et surtout chaque détecteur. En effet, une tuile traitée dans la chaîne est potentiellement acquise par plusieurs détecteurs. Ces images sont ramenées à la résolution réduite des produits L2 en interpolant les valeurs d'angles au niveau des zones de recouvrement des

détecteurs. L'image raster des angles de visée est finalement reconstituée à l'aide d'un masque associant chaque pixel à chaque détecteur de la tuile.

Pour simplifier les calculs à la pleine résolution dans l'algorithme de correction atmosphérique, l'image est divisée, par bande spectrale, en sous zones d'angles de visée constants. Des grilles d'angles de visée par détecteur sont générées et contiennent la valeur moyenne des angles sur chaque zone. Une image à pleine résolution donne pour chaque pixel la zone à laquelle il appartient.

Il est important de noter que si les grilles contiennent déjà des coordonnées physiques, il n'est plus nécessaire de multiplier les valeurs d'angles par la taille du pixel au niveau de chaque algorithme.

5.8.1.1.3 Module SubSampling Sentinel-2

Images de réflectance

Lors de la lecture d'un produit Sentinel-2 de niveau 1, les images de réflectance sont scindées en trois images contenant chacune les bandes de résolution identique (10m, 20m, 60m). Ces images sont rééchantillonnées à la résolution réduite des produits L2 puis reconcaténées dans une vecteur image unique.

Masque des bords

Dans le cas Venμs et Sentinel-2, un masque des bords est généré à partir de l'image de réflectance de niveau 1 en déclarant comme pixels de bords d'image tous les pixels à No_Data dans le produit de niveau 1.

Masque des pixels saturés

Pour Sentinel-2, les pixels saturés sont contenus dans des fichiers GML sous forme de polygones. Des fonctionnalités présentes dans GDAL permettent de convertir des polygones en une image raster.

Deux masques de pixels saturés sont fournis dans le header principal d'un produit Sentinel-2 1C :
- Un masque des pixels saturés « bord » et
- Un masque des pixels saturés radiométriques.

Une fois les images raster produites, un *ou* logique est réalisé entre les deux masques. Ces masques sont ensuite fournis au filtre Subsampling pour être ré échantillonnées aux différentes résolutions.

5.8.1.2 Note sur la gestion de la multi-resolution

La correspondance des bandes spectrales avec leurs indices informatiques, initialement définies dans le GIP_L2COMM, est ajoutée dans la factory de chaque capteur. Ainsi, une table de correspondance d'indice est définie spécifiquement pour Sentinel-2 afin de faire le lien entre les bandes thématiques (B01, B02, . . .) et leurs indices informatiques en fonction de la résolution :
- B02 => indice 1 pour la résolution 10m,
- B03 => indice 2 pour la résolution 10m,
- B05 => indice 1 pour la résolution 20 m,
- etc.

Ces tables d'indices sont utilisées pour identifier les bandes en fonction des deux images de type « VectorImage » utilisées pour chacune des deux résolutions.

La classe python **BandInformations** contient une table de correspondance associant l'indice informatique dans l'image au code de la bande. Dans le cas Sentinel-2, cette classe contient les deux tables de correspondance pour les résolutions 10m et 20m. Une structure est définie pour associer à chaque bande spectrale d'une résolution donnée :
- son indice dans un produit complet comportant les 13 bandes spectrales : tabR10[3].index = 7,
- la chaîne de caractère associée à cette bande : tabR10[3].bandcode = « B07 ».

Cette classe offre également un ensemble de « services » tels que le renvoi de l'indice à partir de l'indice informatique de la bande dans l'image de résolution associée, le renvoi de l'indice à partir du code ou l'inverse.

5.8.2 Composants IO

5.8.2.1 Gestion des LUTs dépendantes des bandes spectrales

Une fois encore, les contraintes multi capteurs et multi résolutions ont des impacts sur la gestion des LUTs au niveau de la classe python L2Processor.

Une LUT est définie par capteur et par bande spectrale. Cela implique que les fichiers GIPPs associés sont dépendants du satellite et que la chaîne peut, au cours d'un même traitement, gérer plusieurs LUTs : une LUT pour le produit à la date D (landsat 5 par exemple) et une autre pour le produit à la date D-1 (qui peut être un produit Landsat 7). Au lieu de registrer un seul fichier GIPP par traitement, la chaîne L2 doit maintenant travailler avec autant de fichiers GIPPs que de produits traités.

De plus, comme la résolution L2 dépend de la bande spectrale dans les produits Sentinel-2, les traitements lancés à la pleine résolution L2 sont séparés de manière à ne traiter qu'une seule résolution à la fois. La liste des fichiers associés à un plugin est établie en utilisant la table de correspondance entre les bandes spectrales et les résolutions. Par exemple, pour chaque bande spectrale *bs* à la résolution de 10m, le fichier LUT correspondant est chargé (table_correspondance[bs] = indice de la bande parmi les 13 bandes d'un produit sentinel-2, cf section *Note sur la gestion de la multi-resolution*).

Une dernière spécificité des produits Sentinel-2 doit être prise en compte lors de la génération des LUTs réduites. La variation des angles de visée dans le champ ne peut plus être négligée à la pleine résolution des produits L2. L'image est divisée en zones d'angles de visée constants. Une **LUT réduite est générée par zone** en fixant les valeurs des angles zénithaux solaire et de visée et de l'angle azimutal relatif.

Finalement, sont créés autant de vecteurs de **LookUpTable** contenant une bande par zone, que de résolutions L2 présentes dans le produit traité.

5.8.3 I3D

I3D est une librairie C++ permettant de calculer l'intersection d'une direction avec un MNT en utilisant :
- Un Mnt,
- Un point de départ,
- Un vecteur de visée.

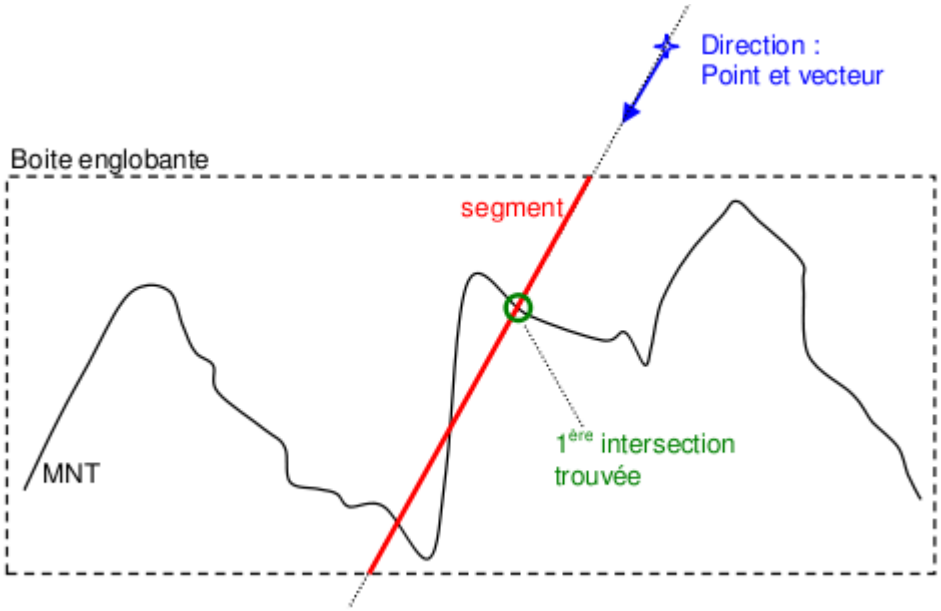
La librairie a été modifiée afin de s'intégrer au pipeline OTB notamment en permettant d'injecter le MNT sous forme de buffer.

L'algorithme commence en définissant une boite englobante à partir de l'altitude min et max du MNT ainsi que des limites de la zone chargée (typiquement le nombre de ligne et colonne en repère capteur).

Ensuite les différents points d'intersection entre la ligne de visée et les plans du cube sont listés. Ces points représentent des points d'entrée du rayon qui permettront ensuite d'établir l'intersection. Ces points sont ensuite filtrés. Avant l'évolution FT-2168 ces points subissaient non seulement un test d'inclusion dans le cube mais également un test d'égalité. Cependant le test d'égalité a été enlevé car de toute façon seul le point le plus pertinent est conservé.

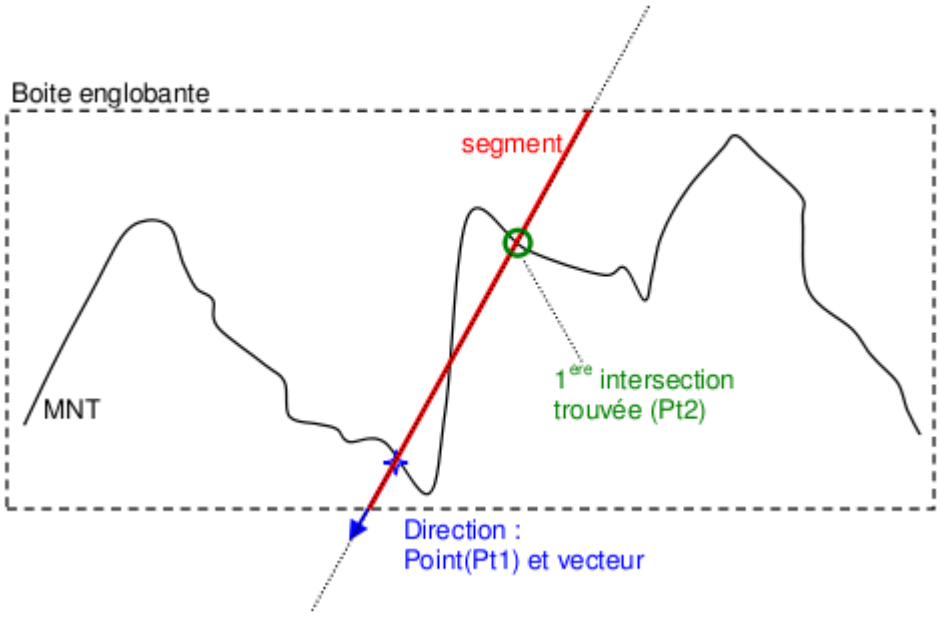
Après filtrage le nombre de points trouvés permet de déterminer le cas de figure :
- 0 point : le rayon ne passe pas par le cube, aucune intersection possible,
- 1 point : le rayon est tangent au cube, aucune intersection possible,
- 2 points et plus : avant la FT-2168 lorsque plus de 2 points étaient trouvés le programme sortait en erreur et ne cherchait pas plus loin. Cependant bien qu'au sens mathématique il ne puisse y avoir plus de deux points il se trouve que les imprécisions du modèle numérique permettent de trouver 2 points très proches mais pas égaux au sens informatique sur les arrête du cube ce qui résultait en une sortie prématurée du programme alors que les conditions étaient valides. Dorénavant dès qu'au moins deux points sont trouvés dans le cube la recherche d'intersection continue.

Le segment définis par le point d'entrée et la direction de visée est alors parcouru pour déterminer l'intersection avec le MNT.



Une autre utilisation de l'algorithme permet de déterminer si un point est caché ou dans l'ombre d'un relief. Pour cela, le point fourni à l'algorithme est le point que l'on souhaite analyser, la direction de visée étant celle du capteur.

Si l'algorithme trouve une intersection avec le MNT différente du point d'entrée alors le point est caché.



CHAPITRE 6

Les algorithmes

6.1 Séquencement des étapes de la chaîne L2

Le développement de l'outil MAJA ne remet pas fondamentalement en cause l'enchaînement des algorithmes de la chaîne VENµS L2.

Le câblage de la chaîne L2 est remanié pour prendre en compte les spécificités algorithmiques de chaque capteur :

— Quelques algorithmes, tels que la détection des cirrus ou de la neige, sont spécifiques à un capteur donné et nécessitent un câblage particulier. Des paramètres définis dans le fichier de configuration permettent d'activer ou pas un algorithme en fonction du capteur considéré.

— Le traitement à plusieurs résolutions L2 pour un même produit Sentinel-2 entraîne des modifications dans l'enchaînement des algorithmes de la chaîne existante.

Des remaniements étant déjà opérés dans la classe *vnsL2Processor*, la duplication des appels aux algorithmes *Cloud Masking*, *Water Masking* et *Scattering Correction* en fonction du mode de traitement (Init ou Nominal) est supprimée. Pour une meilleure lisibilité dans le code, le câblage des données ou le paramétrage des algorithmes spécifiques à un mode donné sont encapsulés dans des tests.

La gestion des résolutions multiples n'apparaît pas au niveau de l'enchaînement des algorithmes si ce n'est dans le passage des images d'entrée aux différents filtres. Les appels multiples aux différentes résolutions L2 sont gérés à l'intérieur même des filtres algorithmiques de manière à ne pas remonter dans la chaîne principale des traitements nécessaires à la génération d'un produit intermédiaire validé dans les tests de validation algorithmiques (TVA).

La lecture ou l'écriture des produits à résolutions multiples sont prises en charges par le plugin dédié.

Les évolutions apportées dans la version MACCS 4.0 ne modifient pas l'enchainement des algorithmes dans la chaine L2.

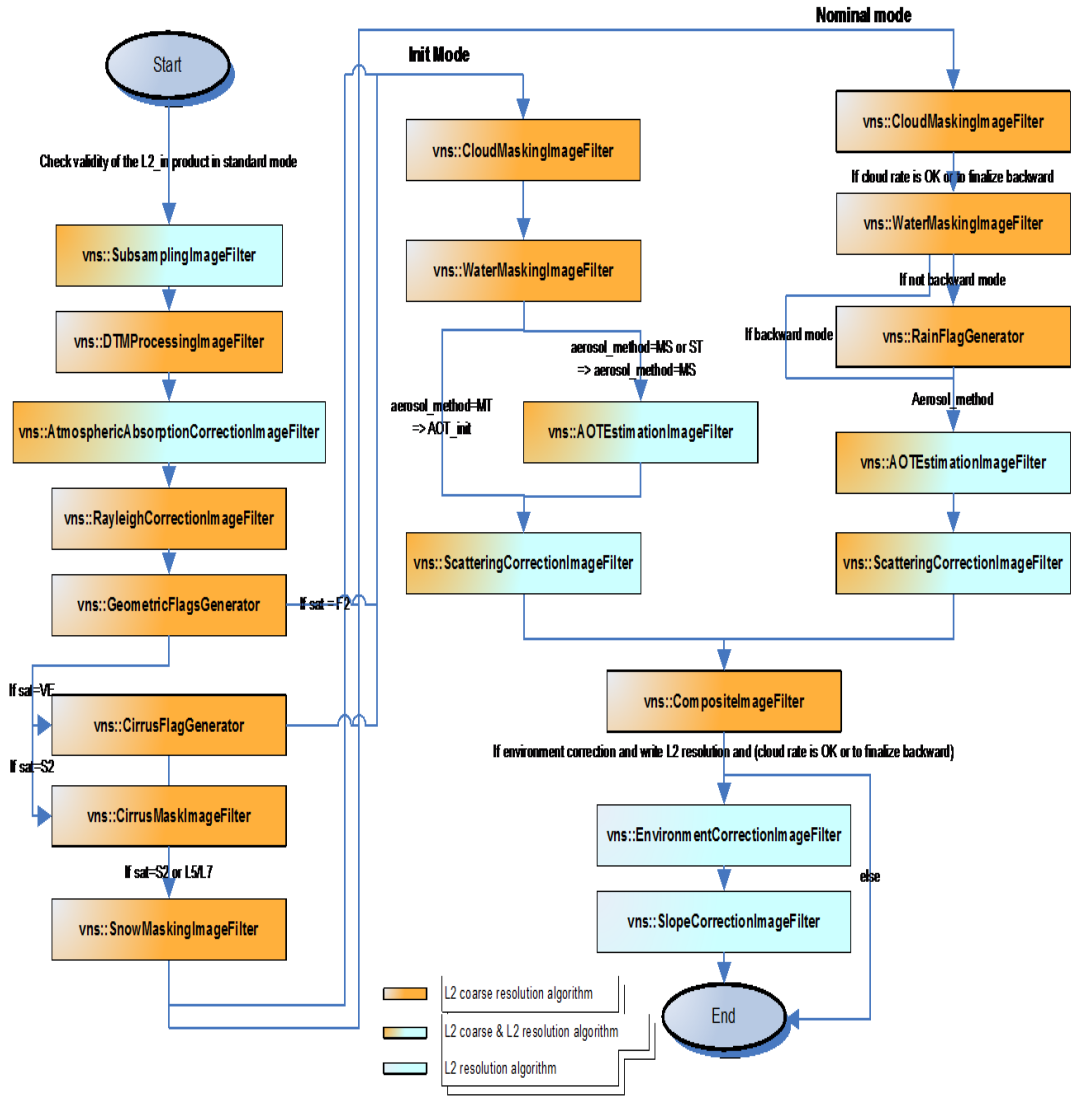


Figure 8 : Séquencement des étapes de la chaîne L2

6.2 Arborescence du code source

Le répertoire des algorithmes est organisé comme suit :

Le répertoire « **Algorithms** » contient le code source des algorithmes implémentés dans MAJA. Ce répertoire est constitué des sous répertoires suivants :

- Le répertoire “Common” contient des filtres de base utilisés par plusieurs traitements algorithmiques de haut niveau,
- Le répertoire « L2 » contient les algorithmes des chaînes images, répartis dans des sous répertoires :
  - “AOTEstimation”
  - “CirrusFlag”
  - “CloudMasking”
  - “CompositeImage”
  - “EnvironmentCorrection”
  - “RainDetection”

— “ScatteringCorrection”
— “SnowMasking”
— “WaterMasking”
— “AtmosphericAbsorptionCorrection”
— “CirrusMask”
— “DTMProcessing”
— “GeometricFlags”
— “RayleighCorrection”
— “SlopeCorrection”
— “Subsampling”

6.3 Les composants logiciels

6.3.1 Démarche générale d’implémentation

Les traitements algorithmiques de la chaîne VENµS L2 sont modifiés pour prendre en compte les produits Sentinel-2 (Level 1-C).

De manière générale, les algorithmes de la chaîne MAJA doivent supportés deux nouvelles contraintes. Ils doivent être à la fois :

— **multi capteurs** pour pouvoir traiter des produits L1 VENµS et Sentinel-2 et
— **multi résolutions** pour supporter les produits Sentinel-2 (level 1-C, S2A et S2B) dans lesquels les bandes spectrales ont des résolutions différentes.

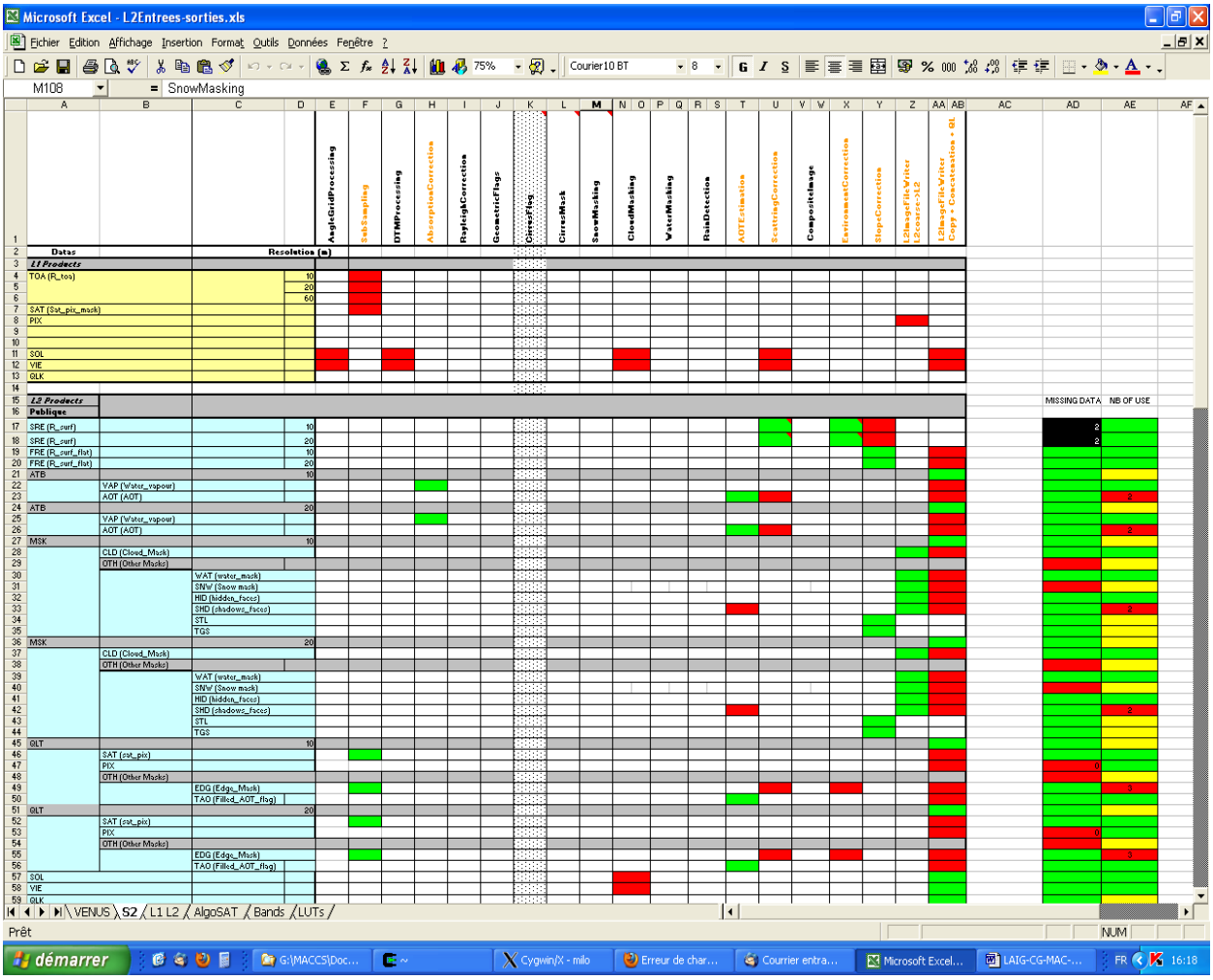
De plus, lors du traitement des produits Sentinel-2, certains algorithmes travaillant à la pleine résolution des produits L2 (correction de la diffusion) doivent maintenant prendre en compte la **variation des angles de visée dans l’image**.

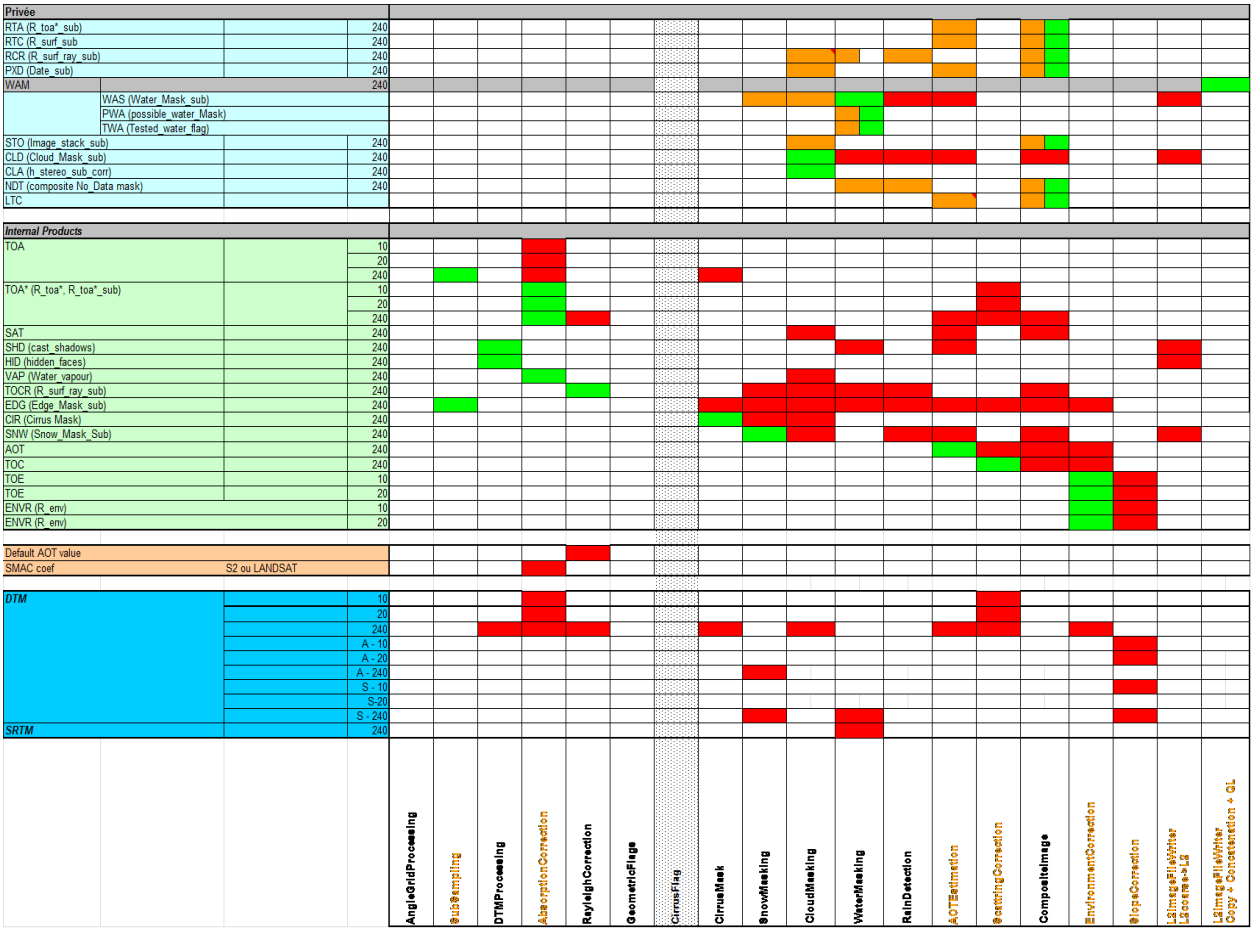
Enfin, de nombreuses évolutions sont apportées à l’**algorithme d’estimation de l’épaisseur optique des aérosols** avec l’ajout de **deux nouvelles méthodes** : la méthode multi-spectrale et la méthode spectro-temporelle.

Il est important de noter que l’ajout ou le remaniement d’algorithmes entraîne une mise à jour du GIPP L2COMM contenant les paramètres de traitement (voir section *Interface des GIPPs*).

Avant de détailler les évolutions dans chaque algorithme, il convient d’analyser l’impact des contraintes multi capteurs et multi résolutions sur l’ensemble des algorithmes de manière à définir différents choix d’implémentation réalisables dans la chaîne L2.

Dans un premier temps, le travail d’analyse mené pour VENµS sur la circulation des données dans la chaîne a été repris pour les produits Sentinel-2 de manière à identifier les nouvelles entrées/sorties de la chaîne et les algorithmes concernés par la gestion de plusieurs résolution L2.


Microsoft Excel - L2Entrees-sorties.xls
Fichier Edition Affichage Insertion Format Outils Données Fenêtre ?
M108
SnowMasking
AngleGridProcessing
SubSampling
DTMProcessing
AbsorptionCorrection
RayleighCorrection
GeometricFlags
CirrusFlag
CirrusMask
SnowMasking
CloudMasking
WaterMasking
RainDetection
AOTEstimation
ScatteringCorrection
CompositeImage
EnvironmentCorrection
SlopeCorrection
L2ImageFileWriter L2coarse->L2
L2ImageFileWriter Copy + Concatenation + GL
Dates
Resolution (m)
L1 Products
TOA (R_toa)
SAT (Sat_pix_mask)
PIX
SOL
VIE
GLK
L2 Products
Publiques
SRE (R_surf)
FRE (R_surf_flat)
ATB
VAP (Water_vapour)
AOT (AOT)
MSK
CLD (Cloud_Mask)
OTH (Other Masks)
WAT (water_mask)
SNW (Snow mask)
HID (hidden_faces)
SHD (shadows_faces)
STL
TGS
QLT
SAT (sat_pix)
PIX
OTH (Other Masks)
EDG (Edge_Mask)
TAO (Filled_AOT_flag)
MISSING DATA
NB OF USE
VENUS S2 L1 L2 AlgoSAT Bands LUTs
Prêt
NUM
démarrer
16:18


Privée
RTA (R_toa*_sub) 240
RTC (R_surf_sub 240
RCR (R_surf_ray_sub) 240
PXD (Date_sub) 240
WAM 240
WAS (Water_Mask_sub)
PWA (possible_water_Mask)
TWA (Tested_water_flag)
STO (Image_stack_sub) 240
CLD (Cloud_Mask_sub) 240
CLA (h_stereo_sub_corr) 240
NDT (composite No_Data mask) 240
LTC
Internal Products
TOA 10
20
240
TOA* (R_toa*, R_toa*_sub) 10
20
240
SAT 240
SHD (cast_shadows) 240
HID (hidden_faces) 240
VAP (Water_vapour) 240
TOCR (R_surf_ray_sub) 240
EDG (Edge_Mask_sub) 240
CIR (Cirrus Mask) 240
SNW (Snow_Mask_Sub) 240
AOT 240
TOC 240
TOE 10
TOE 20
ENVR (R_env) 10
ENVR (R_env) 20
Default AOT value
SMAC coef S2 ou LANDSAT
DTM 10
20
240
A - 10
A - 20
A - 240
S - 10
S-20
S - 240
SRTM 240
AngleGridProcessing
SubSampling
DTMProcessing
AbsorptionCorrection
RayleighCorrection
GeometricFlags
CirrusFlag
CirrusMask
SnowMasking
CloudMasking
WaterMasking
RainDetection
AOTEstimation
ScatteringCorrection
CompositeImage
EnvironmentCorrection
SlopeCorrection
LaImageFileWriter
LaImageFileWriter
Copy + Concatenation + GL

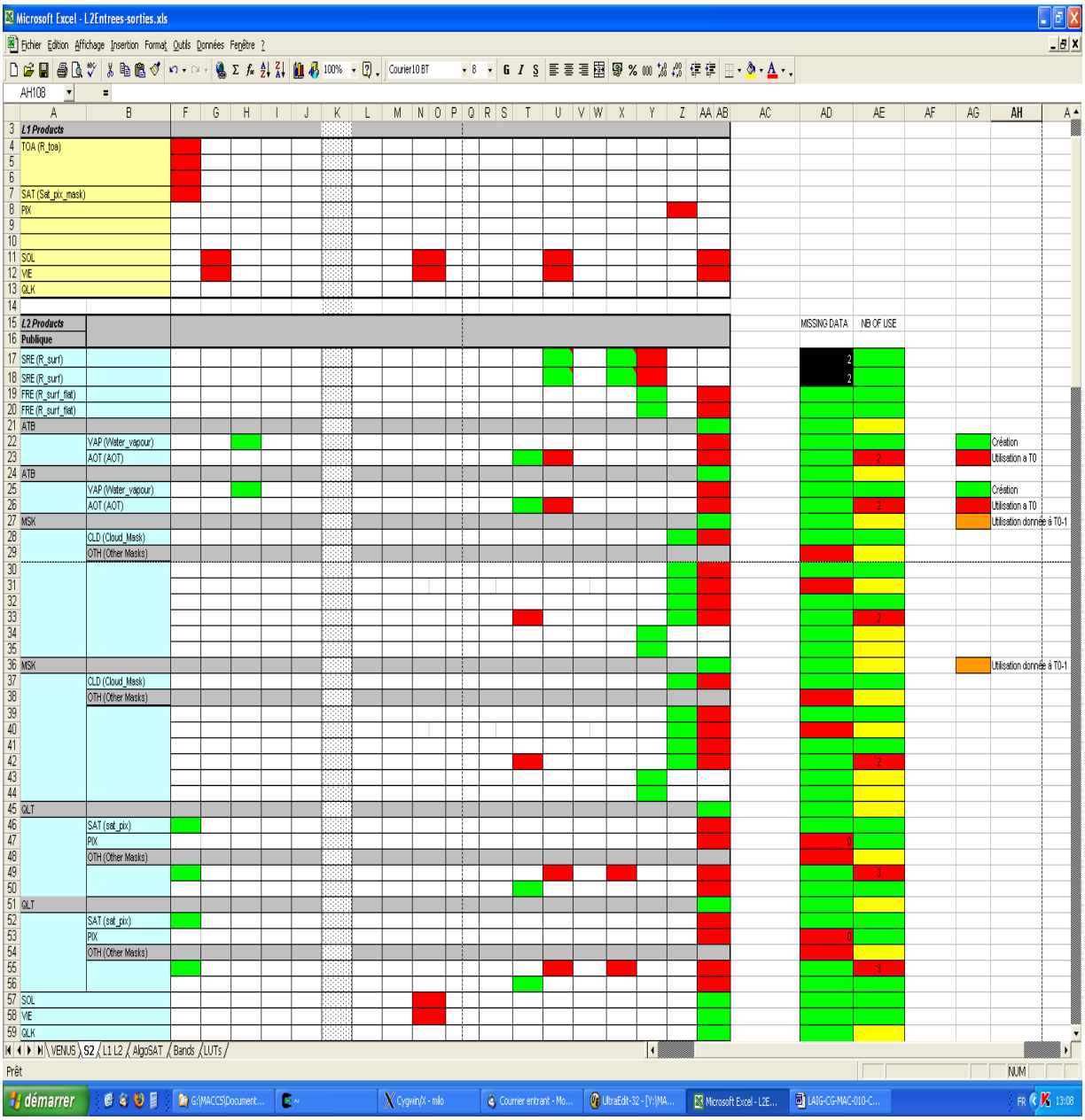


Figure 10 Circulation des données dans la chaîne L2 pour un produit Sentinel-2

Pour gérer la composante multi capteurs dans la chaîne, différents choix d'implémentation ont été identifiés :

- Câblage des algorithmes spécifiques à un capteur via des paramètres de test ajoutés dans le fichier de configuration,
- Combler l'absence d'un plan image en entrée d'un algorithme par la génération d'une image constante :
- soit au moment de lire les données de niveau 1 comme c'est le cas pour le plan contenant l'altitude des nuages (ce qui est totalement transparent dans l'enchaînement des algorithmes),
- soit lors du câblage des algorithmes via des paramètres de test (génération d'une image de label des nuages avec un label unique pour toute l'image),
- Ajouter des entrées optionnelles aux algorithmes. Cela implique que chaque accès à cette entrée est géré par un test d'existence de cette donnée. (la génération de l'image composite utilise la partie privée d'un produit L2 à la date D-1 dans le cas nominal mais pas en mode init).

Chaque solution présente des avantages et des inconvénients. Par exemple, l'utilisation d'entrées optionnelles dans les

couches basses des algorithmes impliquent soit l'ajout des tests dans des boucles sur les pixels à chaque fois que l'on veut manipuler la donnée soit de dupliquer le code. En revanche, cela est tout à fait envisageable quand ces données sont utilisées dans des filtres dits « composite » qui ne change pas l'implémentation de l'algorithme appelé.

La génération d'image constante permet, quant à elle, de conserver le code le plus générique possible pour tous les capteurs tant que cela n'impacte pas les performances de la chaîne (à la résolution réduite L2 par exemple). Un compromis entre garder la généricité du code et lancer des algorithmes qui ne donneront aucun résultat significatif doit être trouvé. C'est pourquoi, il n'est pas possible de définir une ligne de conduite unique pour tous les algorithmes présents dans la chaîne. Une analyse des évolutions algorithmiques de la chaîne L2 est nécessaire pour choisir au cas par cas la solution la plus appropriée.

6.3.2 Description des evolutions algorithmiques de la chaine

Dans ce chapitre, seuls les traitements algorithmiques concernés par les évolutions MAJA sont présentés.

Note pour la version 3.0 : les spécificités liées aux satellites Formosat et Landsat sont présentées même si ces capteurs (i.e. les plugins) correspondants ne sont pas traités dans la version 3.0. Ils le seront dans la prochaine version.

Note pour la version 4.0 : les plug-ins des capteurs Formosat, Landsat 5, 7 et 8, et Spot4 ont été développés. Des évolutions algorithmiques (décrites dans la version 2.2 du document LAIG-ST-MAC-20-CNES_SpecMethode-Trace) ont également été apportées entrainant la mise à jour de certains GIPPs et notamment du GIPP L2COMM. Les algorithmes concernés sont :

- Cirrus Mask
- Snow Masking
- Cloud Masking
- Water Masking
- AOT Estimation

  Note pour la version MAJA 1.0 : la méthode de détection des nuages de CloudMasking a été modifiée pour intégrer la méthode du DLR. De plus la correction atmosphérique a été modifiée pour ajouter en post processing l'estimation de la vapeur d'eau. Le GIPP L2COMM a été modifié afin d'ajouter les nouveaux paramètres nécessaires.

  Note pour la version MAJA 4.0 : la méthode de détection des nuages de CloudMasking a été modifiée pour intégrer la méthode d'Olivier Hagole. Le GIPP L2COMM a été modifié afin d'ajouter les nouveaux paramètres nécessaires.

  Les algorithmes concernés sont :
- Correction of Atmospheric Absorption

  Note pour la version MAJA 3.0 : L'intégration des données CAMS au sein de MAJA permet maintenant une bien meilleure correction en fonction de la zone géographique.
- Cloud Masking

  Note pour la version MAJA 2.0 : Des évolutions algorithmiques ont été apportées à cette version concernant la correction des effets directionnels d'une part et la correction des nuages cirrus d'autre part. Des modifications du GIPP L2COMM ont découlé de ces évolutions.

  Les Algorithmes ajoutés sont donc :
- Cirrus Correction
- Directional correction

6.3.2.1 AngleGrid processing

Ce module a été ajouté dans le [DA02] pour spécifier le calcul des angles solaires et de visée en fonction des différents capteurs. De manière générale :

— les angles de visée sont constants pour les satellites Landsat et Formosat mais dépendent de la position du pixel et de la bande spectrale dans le cas Venμs et Sentinel-2,
— les angles solaires sont constants pour les capteurs Landsat et Formosat mais dépendent de la position du pixel pour Venμs et Sentinel-2 (mais non de la bande spectrale).

Dans la chaîne VENμS L2, la génération des grilles d'angles n'est pas réalisée dans un module spécifique mais au niveau de la « Factory » chargée de la lecture du produit de niveau 1 en fonction de chaque capteur.

6.3.2.2 DTM Processing

Peu de changement sont réalisés dans cet algorithme si ce n'est la prise en compte des différentes grilles d'angles solaires et de visée en fonction du capteur.

Dans la chaîne VENμS L2, le filtre attend deux grilles d'angle de visée en entrée (VIEB5 et VIEB6), ces dernières étant disponibles dans un produit Venμs. Pour s'adapter à Venμs, deux grilles identiques sont créées pour Formosat avec une valeur d'angle constante sur toute l'image.

Pour adapter ce filtre aux produits Sentinel-2 et Landsat, une image multi bandes, où chaque paire de bandes (grilles X et Y) correspond à un plan (B5, B6, . . .), est générée lors des prétraitements (au niveau de la « Factory » qui gère la lecture des produits de niveau 1) et fournie en entrée du filtre. Cette solution permet de garder le code le plus générique possible.

6.3.2.3 Reduced Lut Computation

Ce module a été ajouté dans la version 2.0 de MAJA bien que l'algorithme d'extraction de la mini-lut à partir de la LUT globale était disponible avant. Cependant l'ajout de la correction des effets directionnels qui s'applique directement sur cette mini-lut a justifié que cet algorithme soit désormais une boite à part entière.

Il s'agit ici d'extraire par interpolation sur les angles solaires et les angles de vues du produit une sous-lut propre au produit qui va ensuite être utilisé dans les algorithmes suivant. On obtient alors une lut à 3 dimensions alors que la lut d'entrée est,elle, à 6 dimensions.

La correction des effets directionnels consiste quant à elle à appliquer pour chaque bande des coefficients multiplicateurs préalablement calculé par l'algorithme. Cette correction permet de simuler un produit ayant été acquis avec un angle différent de celui réel afin de pouvoir faire de multi temporel avec deux produit ayant été pris avec des angles légèrement différents. Cela est notamment le cas sur SENTINEL2 avec deux acquisitions de la même zone sur deux orbites différentes.

A partir de MAJA v3.0, ce module a également la responsabilité, sous réserve d'activation et de disponibilité de la donnée exogène, de générer les LUTs DIFT, DIRT, ALBD et TOCR à partir des données CAMS indiquant les teneurs de l'atmosphère relativement à certains aérosols, ainsi que des LUTs relatives à ces même aérosols. Il est dorénavant possible de d'estimer, grâce aux données CAMS, des lut DIFT, DIRT, ALBD et TOCR plus représentatives de la réalité physique lors de l'acquisition en lieu et place des LUTs constantes utilisées par défaut. En contrepartie, les données auxiliaires à fournir à l'application MAJA sont plus nombreuses et ont une période de validité contraignante, c'est pourquoi la capacité d'utiliser des LUTs constantes est toujours disponible et utilisée dans le cas général.

6.3.2.4 Subsampling

Modifier le module actuel « Susampling » afin qu'il gère l'aspect multi résolution de S2 n'est pas la solution retenue. La solution consiste à utiliser ce module pour les cas VENµS, Formosat et Landast. Pour le cas Sentinel-2, on développe un nouveau module SubSamplingS2, en évitant autant que faire se peut la duplication de code.

Par ailleurs, l'appel au module SubSampling est déplacé dans les Factory afin de regrouper les spécificités liées aux formats des produits L1 dans le Plugin du capteur dédié.

A partir de la version 4.2, les étapes de rééchantilonnage sont réalisées par un module commun à tout MACCS, la classe vnsPadAndResampleImageFilter. Elle s'appuie sur des filtres d'interpolation de l'OTB, d'ITK et certains développés dans MACCS pour des soucis de performances.

Le filtre de l'OTB StreamingResampleImageFilter initialement utilisé pour réaliser les rééchantillonnages a été remplacé à partir de la version 4.8 par le filtre vnsGridResampleImageFilter, beaucoup plus performant en temps d'exécution, pour des résultats numériques identiques.

6.3.2.5 Correction of Atmospheric Absorption

Quelques modifications sont apportées à l'algorithme de correction atmosphérique par rapport à celui utilisé dans la chaîne VENµS L2.

Tout d'abord, il est important de noter, même si cela n'a pas d'impact au niveau de l'algorithme lui-même, que les coefficients SMAC sont dépendants du satellite S2-A, S2-B, L5, L7. En effet, les coefficients SMAC sont lus dans des GIPPs. La chaîne utilise le fichier GIPP associé au produit traité.

D'autre part, le contenu en vapeur d'eau n'est plus fonction du rapport entre les réflectances à 910 et 865 nm (cas Venµs) mais entre les bandes 940 nm et 865 nm (pour Sentinel-2). Il convient d'utiliser un paramétrage générique adapté à tous les capteurs.

La routine smac_inverse (fournie par le CNES) est maintenant dépendante des angles de visée. Cette routine, ayant été ré implémentée par CS dans la chaîne VENµS L2, est mise à jour.

La principale évolution de cet algorithme est liée à la prise en compte des différentes résolutions d'un produit Sentinel-2. Dans la chaîne VENµS, ce filtre est appelé deux fois, une fois à la résolution réduite L2 et une fois à la pleine résolution. L'image du contenu en vapeur d'eau est générée, à chaque résolution L2, à partir d'une LUT en fonction du rapport des réflectances à 910 et 865 nm. Dans un produit Sentinel-2, ces bandes sont à des résolutions L2 différentes et donc contenues dans des vecteurs images différentes. Ce filtre est donc remanié. Il génère l'image du contenu en vapeur d'eau à la résolution réduite L2 et la ré échantillonne aux différentes résolutions L2 avant d'appliquer les corrections gazeuses à une ou plusieurs images de réflectance (10m et 20m pour Sentinel-2).

A partir de MAJA 1.0, l'estimation de la vapeur d'eau est post-processée afin de limiter les erreurs d'estimation sur les zones sombres. Elle est notamment « bouchée » à l'aide d'un algorithme de « gap filling » et completée d'un masque indiquant si la teneur en vapeur d'eau à été estimée ou bouchée pour chaque pixel.Cirrus Flag

Aucune modification n'est apportée par rapport à la chaîne VENµS L2. Cependant, il est important de noter que cet algorithme est désormais spécifique à VENµS. En effet, le plan d'altitude des nuages n'est pas disponible dans les produits L1 Formosat et Landsat et une autre méthode de détection des cirrus est définie pour Sentinel-2.

Pour uniformiser la logique choisie pour les algorithmes de détection des cirrus et de neige, qui sont spécifiques à chaque capteur, un paramètre est ajouté au fichier de configuration de manière à activer ou non cet algorithme en fonction du capteur traité.

6.3.2.6 Cirrus Masking

Répond à : ST-169-CirrusMaskS2-0010 dans : DA02 Conformité : C
Répond à : ST-169-CirrusFlagS2-0020 dans : DA02 Conformité : C
Répond à : ST-169-CirrusFlagS2-0030 dans : DA02 Conformité : C

Les produits Sentinel-2 possèdent une bande spectrale à 1,38 µm qui a une très forte absorption à la vapeur d'eau. De ce fait, un simple seuillage sur la valeur de la réflectance TOA de cette bande permet de détecter un pixel nuageux de type cirrus. Cependant, la projection des ombres des nuages restant délicate, l'indicateur global « cirrus_flag » est conservé.

Cet indicateur est levé lorsque le pourcentage de pixels nuageux de type cirrus est supérieur à un seuil.

Cette classe est spécifique aux produits Sentinel-2 et génère en sortie un masque des pixels de type cirrus. Un paramètre permettant d'activer ou pas cet algorithme est donc ajouté dans le fichier de configuration.

Par ailleurs, ce masque étant réutilisé dans les algorithmes Snow Masking et Cloud Masking, une image nulle est générée dans L2Processor pour les autres capteurs de manière à conserver le code le plus générique possible. En effet, il est préférable, dans ce cas, de produire une image constante plutôt que de gérer des entrées images conditionnelles. L'utilisation d'entrées optionnelles modifie le cœur des méthodes pour l'adapter aux entrées fournies et peut entraîner des duplications impropres de code en fonction du capteur.

Dans la version 4.0, les modifications apportées à cet algorithme concernent tous les capteurs. Le seuil sur la réflectance 1.38µm au-delà duquel un pixel est déclaré nuageux n'est plus un simple paramètre GIPP mais est calculé à partir d'une équation linéaire en fonction de l'altitude (seuil_reflectance = gain * altitude +offset).

Le MNT (à la résolution réduite des produits L2) est donc ajouté en entrée de l'algorithme.

Deux nouveaux paramètres GIP_L2COMM sont créés pour remplacer le paramètre Cirrus_Refl_Threshold : Cirrus_Mask_Threshold_Offset et Cirrus_Mask_Threshold_Gain.

6.3.2.7 Snow Masking

L'algorithme de détection de la neige est mis en place dans la chaîne MAJA pour les capteurs Sentinel-2, Landsat et Venus. Les capteurs Sentinel-2 et Landsat disposant d'une bande spectrale SWIR (Short-Wave Infra-Red) ils peuvent l'utiliser afin de générer ce masque. L'algorithme concernant Venus est plus simple et n'utilise pas de bande SWIR car non présente sur le capteur mais utilise une bande proche afin d'approximer le même comportement. En effet bien que non SWIR la bande rouge absorbe partiellement les longueur d'ondes SWIR et permet donc la détection de neige.

La détection de la neige sur les images Sentinel-2 repose sur un seuillage de l'indice NDSI définit par :

$$NDSI = \frac{\rho_{green} - \rho_{SWIR}}{\rho_{green} + \rho_{SWIR}}$$

La réflectance de la bande SWIR (bande rouge pour Venus) permet de distinguer nettement la neige, qui a une réflectance faible dans cette bande, des nuages qui ont eux une réflectance élevée. De ce fait, les pixels neigeux ont un indice NDSI élevé tandis que les nuages ont un indice faible tendant vers zéro. Un Seuillage sur cet indice permet donc de séparer les pixels neigeux des pixels nuageux.

L'eau pouvant causer des fausses détections, un seuillage supplémentaire dans le rouge permet de palier ces fausses détections.

Un filtrage est également effectué sur les bords de l'image.

Enfin, une opération de morphologie mathématique de type fermeture (dilatation + érosion) est appliquée au masque de neige.

Le masque de neige est utilisé dans de nombreux algorithmes (Cloud Masking, Rain Detection, AOT Estimation, Composite Image). Le câblage de cet algorithme dans L2Processor est réalisé via un paramètre présent dans le fichier de configuration.

Comme décrit ci-dessus, le calcul du NDSI (Normalized Difference Snow Index) et les tests permettant de détectés un pixel neigeux étaient réalisés sur les valeurs de réflectance TOA corrigées de l'absorption atmosphérique et de la diffusion sur le rayleigh. Dans cette nouvelle version, ces valeurs de réflectance sont préalablement corrigées des effets de pente au premier ordre :

refl_corr = refl_surf_ray * cos thetaS / cos ThetaI

thetaS : angle zenithal solaire

thetaI : l'angle d'incidence solaire est l'angle entre la normale à la pente et la direction solaire

Le calcul de l'angle d'incidence solaire est réalisé à la résolution réduite des produits L2 (L2 coarse resolution). Ce calcul est déjà présent dans l'algorithme de correction des pentes et pourra être factorisé.

La détection de la neige sur les zones d'eau n'est plus réalisée excepté en mode init et uniquement si le paramètre Use_Water_mask est fixé à vrai.

6.3.2.8 Cloud Masking

La méthode de détection des nuages et de leurs ombres est similaire à celle de la chaîne VENµS L2. Seules quelques évolutions sont à noter :

- La prise en compte du masque de neige lors du seuillage sur la réflectance dans le bleu,
- La prise en compte du masque de neige dans la méthode de seuillage sur la variation de réflectance,
- La prise en compte du masque de cirrus lors de la génération du masque de nuages issus des critères stéréoscopiques et radiométriques,
- Extraire le sous échantillonnage de l'image d'altitude des nuages spécifique à Venµs et le remonter dans L2Processor.

Pour les capteurs Sentinel-2, Formosat et Landsat, une image nulle du plan d'altitude des nuages est créée à la résolution réduite L2 au niveau du DataManager lors de la lecture des produits de niveau 1. L'appel au sous échantillonnage de ce masque consiste, dans ce cas, à une simple recopie de l'image d'entrée. Le masque associé, utilisé dans Cloud Masking, est créé sans distinction pour tous les capteurs.

- La labellisation des nuages lors de l'affinage de l'altitude pour les images Sentinel-2 et Landsat est la même que pour Formosat, c'est à dire un label unique pour tous les nuages de l'image,
- L'altitude des nuages pour les produits Sentinel-2, Formosat et Landsat est constante sur toute l'image et fixée à une valeur d'altitude par défaut,
- Dans l'algorithme de projection des ombres, l'utilisation de la grille VIEB5 d'un produit Venµs n'a plus de sens pour les capteurs Sentinel-2 et Landsat. La bande spectrale de référence pour la direction de visée est spécifique à chaque capteur et paramétrable.

Remarque : la manipulation de ces grilles dans les algorithmes est réalisée en coordonnées « physiques » et non pas en coordonnées « image ». Si cette conversion est réalisée lors de la lecture d'un produit de niveau 1, elle doit être supprimée au niveau des algorithmes.

Les principales évolutions de la version 4.0 de MACCS sont réalisées dans l'algorithme de détection des nuages.

A partir de MAJA 1.0, la détection initiale mono-temporelle des nuage est effectuée à l'aide de la méthode du DLR impactant les seuils appliqués aux différentes bandes afin de décider si un pixel est nuageux ou non.

6.3.2.8.1 Reflectance Threshold

Le seuillage sur la réflectance dans le bleu et les pixels saturés est complété par un seuillage sur la réflectance dans le proche infrarouge et sur le masque d'eau à la date D-1 (après érosion).

La détection de nuages au-dessus de l'eau est améliorée par l'ajout d'un seuillage supplémentaire sur la réflectance. Ce seuil dépend de la présence de glitter ou pas sur l'image.

Deux paramètres GIP_L2COMM sont ajoutés : Blue_reflectance_threshold et PIR_ reflectance_threshold.

6.3.2.8.2 Reflectance Variation Threshold

Une légère modification est apportée sur le calcul du seuil sur la variation de réflectance dans le bleu. Ce seuil est désormais affecté à la valeur minimum entre le seuil calculé en fonction de la durée écoulée entre les deux produits et un seuil maximum fixé (paramètre d'entrée) qui est ajouté au GIP_L2COMM.

Le GIPP Blue_Var_Threshold est remplacé par les paramètres Min_threshold_var_blue et Max_threshold_var_blue.

6.3.2.8.3 Snow mask correction

Un nouvel algorithme de correction du masque de neige est ajouté au module Cloud Masking.

En effet, de fausses détections de neige pouvant apparaître au cœur de nuages épais, il est préférable de supprimer du masque de neige ces zones neigeuses situées à proximité de nuages ou de no_data et de les ajouter au masque de nuages issu du seuillage sur la réflectance.

Pour déterminer ces zones, l'algorithme suivant est mis en place :
- le masque de neige est labellisé puis dilaté.
- pour chaque label, l'algorithme teste si tous les pixels déclarés neigeux uniquement dans le masque de neige dilaté (et pas dans le masque de neige initial) sont également nuageux ou dans les bords de l'image (no_data)
- Si c'est le cas, tous les pixels de ce label (non dilaté) sont désactivés dans le masque de neige et ajoutés au masque de nuages issu du seuillage sur la réflectance.

Le masque de nuages utilisé inclut le masque de nuages déterminé par les seuillage sur la réflectance et le masque d'eau (Reflactance Threshold) et par la détection des variations de réflectance entre la date D et la date D-1.

Le module Cloud Masking renvoie le masque de neige et le masque de nuages corrigé.

L'interface de ce nouvel algorithme est décrite dans le tableau suivant :

	Description
Entrées	
Cloud_mask_sub.refl	Masque des nuages obtenu par seuillage sur la réflectance dans le bleu et le PIR
Cloud_mask_sub.refl_var	Masque des nuages obtenu par seuillage sur la variation de réflectance
Edge_mask_sub	Masque des bords de l'image et des no_data
Snow_mask_sub	Masque de neige
Sorties	
Cloud_mask_sub.refl	Masque des nuages obtenu par seuillage sur la réflectance dans le bleu et le PIR corrigé
Snow_mask_sub	Masque de neige corrigé

6.3.2.8.4 Shadow mask determination

Le masque des ombres est généré à partir d'équations géométriques qui utilisent le masque des nuages, l'altitude des nuages, les angles solaires et de visée et le MNT pour projeter les nuages au sol et déterminer la position des ombres. La position des ombres au sol est donc liée à l'altitude des nuages. Pour améliorer la localisation de ces ombres, un algorithme d'affinement de l'altitude des nuages fondé sur des critères radiométriques a été mis en place dans la chaîne L2. Il procède par itération sur l'altitude des nuages pour rechercher l'altitude qui correspond à la zone d'assombrissement maximum (due aux ombres) entre l'image à la date D et celle à la date D-1.

Seuls les produits de niveau 1 Venμs disposent d'un plan contenant l'altitude des nuages déterminée par stéréoscopie.

A partir de la version **5.0** de MACCS (DM **982**), pour les capteurs différents de Venμs, l'approche est différente. L'absence de donnée d'altitude des nuages nécessite de s'adapter en déterminant une zone dans laquelle des ombres peuvent se trouver. Pour cela, au lieu d'estimer l'altitude des nuages, leur ombre projetée va être calculée pour chaque altitude dans une certaine gamme. Au sein de ces zones, il sera alors appliqué différents tests statistiques relatifs à la radiométrie de ces zones. La détermination des ombres n'est plus itérative et ne permet alors plus d'estimer l'altitude des nuages. Ces opérations sont assurées par la classe vns : :ShadowMaskDeterminationWithoutKnownAltitude. Ainsi, afin de paramétrer ces nouveaux traitements, de nouveaux paramètres ont été ajoutés au fichier GIPP_L2COMM.

6.3.2.9 Water Masking

Deux tests sont ajoutés dans la version MACCS 4.0 :

- Un test sur la sélection des pixels valides qui exclut les pixels pour lesquels la pente est supérieure à un seuil
- Un test supplémentaire sur la réflectance de surface corrigée du rayleigh dans le rouge qui vient s'ajouter au test sur le NDVI.

L'interface de cet algorithme a donc été légèrement modifiée. L'image des pentes fournie avec le MNT à la résolution réduite des produits L2 est utlisée et deux nouveaux paramètres GIP_L2COMM sont créés : Water_red_refl_threshold, Water_slope_threshold.

6.3.2.10 Rain detection

L'algorithme de détection de la pluie prend en compte le masque de neige. Un masque de neige constant et nul est donné en entrée du filtre pour les capteurs Venμs et Formosat.

De plus, il faut remplacer le paramètre NIR_band par Water_Band dans les GIPP.

6.3.2.11 Estimation of Aerosol Optical Properties

D'importantes évolutions sont apportées à cet algorithme. Il existe maintenant trois méthodes pour estimer l'épaisseur optique des aérosols :

- La méthode multi temporelle repose sur la stabilité de la réflectance de surface pour des images du même site acquises à des dates différentes. Il s'agit de la méthode actuellement implémentée dans la chaîne VENμS L2.
- La méthode multi spectrale utilise des relations empiriques entre les bandes spectrales rouge et proche infra-rouge.
- La méthode spectro temporelle combine les deux méthodes précédentes.

Trois nouveaux filtres vnsMultiTemporalComputeAOTImageFilter, vnsMultiSpectralComputeAOTImageFilter et vnsSpectroTemporalComputeAOTImageFilter sont créées. Le filtre présent dans la chaîne VENμS est repris pour les méthodes multi temporelle et spectro-temporelle.

La sélection des pixels pour l'estimation locale de l'épaisseur optique des aérosols est déclinée en deux versions pour les méthodes multi temporelle et multi spectrale. Le masque des pixels neigeux est pris en compte dans les deux cas. Le critère de sélection sur la variation relative de réflectance entre la date D et la date D-1 est remplacé par un seuillage sur la réflectance dans le rouge et sur le NDVI dans la méthode multi spectrale.

La fonction de coût utilisée pour l'estimation de l'épaisseur optique des aérosols est maintenant dépendante de la méthode utilisée. La fonction de coût de la méthode multi temporelle est la même que dans la chaîne VENμS.

Une nouvelle fonction de coût est ajoutée pour la méthode multi spectrale. La méthode spectro temporelle utilise les deux.

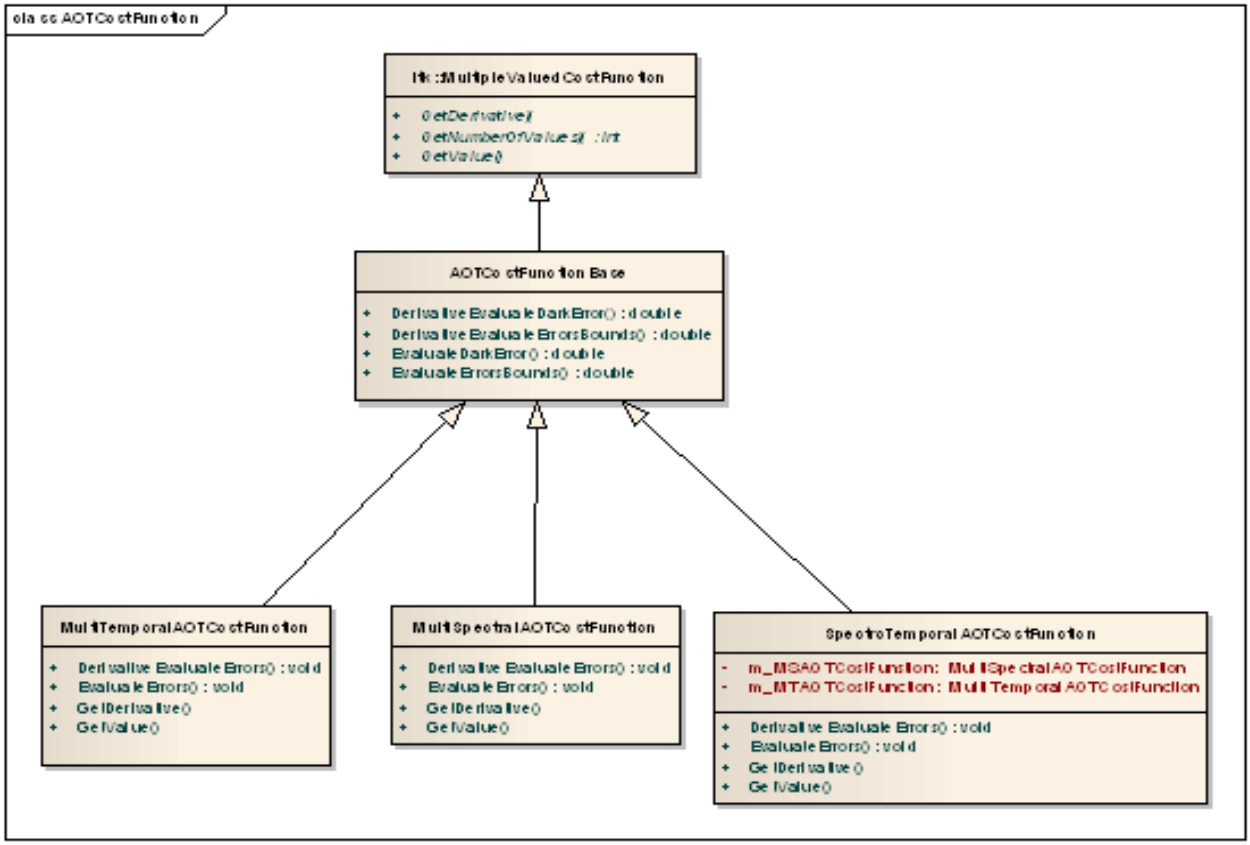


Figure 11 : diagramme de classes pour les fonctions de coût

Un nouveau template (de type enum) est ajouté à la classe vnsAOTEstimation pour définir la méthode à employer. De plus, la partie algorithmique (ré échantillonnage et redécoupage) concernant le passage de la résolution réduite L2 à la pleine résolution pour les images de sortie (AOT et TAO) va être appelée autant de fois qu'il y a de résolutions L2 dans le produit traité.

Seule une évolution a été apportée à cet algorithme dans la version MACCS 4.0 et concerne la sélection des pixels pour la méthode multi temporelle.

Un nouveau seuillage sur la réflectance dans le bleu est ajouté lors de la sélection des pixels utilisés pour le calcul de l'épaisseur optique des aérosols via la méthode multi temporelle. L'objectif est d'éliminer les valeurs de réflectance TOA pour lesquelles l'épaisseur optique des aérosols ne peut être calculée. En effet dans une gamme de réflectance donnée, l'impact de l'épaisseur optique aérosols n'est pas quantifiable sur la mesure de la réflectance.

Pour effectuer ce seuillage, deux seuils encadrant les valeurs de réflectance rejetées sont calculés en recherchant dans une gamme de réflectance (pour des conditions d'acquisition données) les valeurs de réflectance TOA pour lesquelles la différence entre les réflectances de surface obtenues pour deux valeurs extrêmes d'AOT est faible (inférieure à un seuil). Le passage de réflectance TOA aux valeurs de réflectance de surface est réalisé via les mini LUTs calculées pour les conditions d'acquisition (valeurs des angles au centre de l'image) associées au produit traité.

Paramètres	Description
Min_Difference_Thresholds_Calculation	Seuil sur la différence entre des réflectances de surface obtenues pour deux valeurs extrêmes d'AOT
First_AOT	Valeur minimum d'AOT pour calculer la première gamme de réflectance
Second_AOT	Valeur maximum d'AOT pour calculer la première gamme de réflectance
TOA_Reflectance_Min	Valeur minimum de la réflectance TOA pour déterminer la gamme de réflectance
TOA_Reflectance_Max	Valeur maximum de la réflectance TOA pour déterminer la gamme de réflectance
TOA_Reflectance_Step	Pas entre deux valeurs de réflectance dans la gamme.

6.3.2.12 Cirrus Correction

L'algorithme de correction des zones de cirrus a été introduit dans MAJA V2.0. Celui-ci permet de corriger les zones de l'image impactées par les nuages cirrus afin d'améliorer le rendu visuel. Afin de traiter au mieux les images, la distance de Maurer est utilisée afin de pondérer la correction permettant ainsi d'appliquer une correction plus importante au centre des nuages par rapport à leurs bords. Seuls les capteurs disposant d'une bande cirrus peuvent activer la correction cirrus.

6.3.2.13 Scaterring Correction

Dans la chaîne VENμS L2, la variation des angles solaires et de visée dans le champ est négligée lors de la correction de la dispersion atmosphérique à pleine résolution. De ce fait, la LUT ne dépend plus que de la réflectance de surface, de l'épaisseur optique des aérosols et de l'altitude. L'algorithme travaille avec une LUT dite « réduite » pour des valeurs d'angles constantes.

Pour Sentinel-2, la variation des angles dans le champ est trop importante pour être négligée. Les angles sont alors considérés constants par zone dans l'image. Une image contenant la valeur moyenne des angles de visée par zone est créée au niveau du DataManager. La modification consiste donc à prendre en compte cette variation d'angle sur l'image en utilisant une LUT « réduite » par zone. Ces LUTs réduites sont générées dans la classe L2Processor et un vecteur de LUTs (par zone) est passé en entrée de l'algorithme.

De plus, pour gérer les différentes résolutions L2 d'un produit Sentinel-2, deux vecteurs de LUTs réduites sont générés dans L2Processor, l'un contenant les bandes spectrales à 10m et l'autre les bandes spectrales à 20m.

Comme pour l'algorithme de correction atmosphérique, une classe est chargée de l'enchaînement des appels aux classes algorithmiques pures. Elle permet de gérer les traitements aux différentes résolutions en fonction des capteurs. Cette classe attend une entrée par type d'image (réflectance, masque, MNT) et par résolution (résolution réduite, résolutions L2).

6.3.2.14 Composite Image

Il faut maintenant prendre en compte le masque de neige, pour la génération de l'image composite.

Dans la version MACCS 4.0, la recherche des zones à forte épaisseur optique des aérosols à proximité de nuages est supprimée. Toutes les zones à forte AOT sont désormais prises en compte.

6.3.2.15 Corrections for environnement effects

L'algorithme de correction des effets de l'environnement n'est pas modifié par rapport à la chaîne VENµS L2 si ce n'est qu'il faut maintenant gérer les deux résolutions L2 (10 m et 20 m) d'un produit Sentinel-2.

Pour cela, les traitements à la résolution réduite L2 et ceux à la pleine résolution sont séparés dans des filtres distincts :

- le filtre existant génère les images de réflectance d'environnement, de transmission directe et diffuse et d'albedo à la résolution réduite des produits L2,
- un deuxième ré échantillonne les images à la pleine résolution L2, les retaille, supprime les effets de flou sur les bords de l'image de réflectance de l'environnement, génère l'image de transmission totale et applique le terme correctif à la réflectance de surface corrigée de la diffusion.

Ce découpage permet d'appeler les traitements à la pleine résolution autant de fois qu'il y a de résolutions différentes. Les images à la résolution réduite L2 étant générées pour toutes les bandes spectrales, les tables de correspondance entre les bandes spectrales à la résolution « L2 coarse » et celles à la résolution L2 sont utilisées dans le Functor générant l'image de réflectance de surface finale.

Dans le cas Sentinel-2, deux images de réflectance de surface et deux images de réflectance de l'environnement (10 m et 20 m) sont générées en sortie et réutilisées pour la correction des pentes.

6.3.2.16 Slope Correction

Dans l'algorithme de correction des pentes, la problématique multi résolution liée à Sentinel-2 est traitée de la même façon que dans l'algorithme de correction de l'environnement. Les traitements à la résolution réduite des produits L2 et à la pleine résolution sont donc séparés. Les tables de correspondance des bandes spectrales entre les différentes résolutions sont également utilisées.

6.3.3 Description des évolutions des « outils de contrôle et de validation »

6.3.3.1 Évolutions générales

De manière générale, les traitements actuellement disponibles dans l'outil implémenté dans les chaînes L2 VENµS évoluent afin de prendre en compte des produits des niveaux 1, 2 pour les capteurs Sentinel-2 et Landsat.

Le module de lecture de la partie publique d'un produit L2 est modifié pour traiter les données à plusieurs résolutions.

Les évolutions spécifiques sont décrites ci-dessous.

6.3.3.2 Évolutions spécifiques

Seules quelques évolutions spécifiques sont apportées à cet outil :

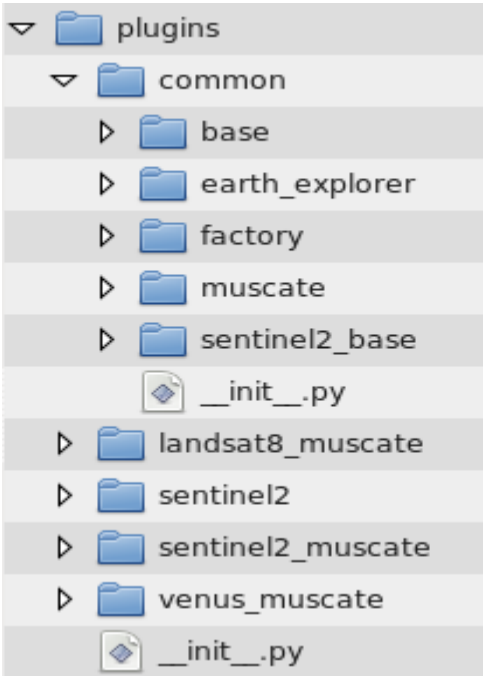
- Pour les cas Sentinel-2 et Landsat, les contours du masque de neige seront soulignés dans le Quicklook généré (dans Exigence 100),
- Un contrôle est réalisé pour vérifier que les 3 canaux sélectionnés par l'utilisateur correspondent bien à des bandes de même résolution (dans Exigence 120),
- L'extraction de pixels sur les plans de résolutions 20 et 60 m d'un produit Sentinel de niveau 1 ou 2 est ajoutée : à partir des coordonnées du pixel central défini dans la résolution 10m, il faut déterminer le pixel équivalent (superposable) pour les plans de résolutions 20 et 60 m (prendre en compte le facteur de résolution sur la taille de la fenêtre à utiliser) (dans Exigence 210).

CHAPITRE 7

Les plugins « capteur »

7.1 Arborescence du code source des plugins

Le code source des plugins est organisé de cette manière :



La base commune **common** n'est pas impactée lors de l'ajout d'un nouveau capteur.

7.1.1 Le répértoire common

Le répértoire **common** contiens les base commune à tous les capteurs ainsi que des base pour chaque famille de produits (EarthExplorer ou Muscate).

Il contient notamment le sous répértoire **base** servant de base à tous les capteurs. Il fournis notamment ces interfaces :

BandsDefinition	Cette classe fournis les informations relative aux bandes du capteurs pour les différents niveaux de produits. Il donne notamment la correspondance entre le nom de la bande et l'index
L1ImageFileReaderBase	Cette classe fournis les images L1 necessaire au traitement. Par exemple toute declinaison capteur de cette classe doit fournir les TOA, EDG, SAT en résolution L2 et coarse.
L2ImageFileReader	Cette classe fournis les images L2 necessaire aux algorithmes de MAJA. Elle doit notamment fournir les images PRIVATE (RTA, RTC, RCR, STO, PXD, NDT, CLD, CLA, WAM and LTC).
L2ImageFileWriter	Cette classe fournis des fonctionalités d'écriture des images du produit L2.
L2HeaderFileWriterProviderBase	Cette classe fournis des fonctionalités d'écriture des métadonnées du produit L2.

Le répértoire **factory** quand à lui contiens les éléments relatifs au système de factory avec notamment :

L1ImageFileReaderProvider	Permet de demander à la factory si un capteur est capable de lire les images du produit L1. Dans ce cas elle retourne une instance du lecteur
L1ImageInformationsProviderFactory	Permet de demander à la factory si un capteur est capable de lire les informations du produit L1. Dans ce cas elle retourne une instance du lecteur
L2HeaderFileWriterProviderFactory	Permet de demander à la factory si un capteur est capable d'écrire les métadonnées du produit L2 et retourne une instance de celui-ci
L2ImageFileWriterProviderFactory	Permet de demander à la factory si un capteur est capable d'écrire les images du produit L2 et retourne une instance de celui-ci
L2ImageFileReaderFactory	Permet de demander à la factory si un capteur est capable de lire les images du produit L2. Dans ce cas elle retourne une instance du lecteur.
PluginActivate	Permet d'enregistré les différents plugins au sein de la factory.

7.1.2 Les répértoires plugins

Chaque plugins est dans son propre répértoire pour plus de clarté. Cependant celui-ci derive obligatoirement soit de la classe de base soit d'une classe derivée propre au capteur (Sentinel2 dérive de Sentinel2_Base). Libre à chaque plugin de fournir un object traitant tout ou partie des produits. On peux imaginer que certains formats n'aient pas pour but d'être écris en L2 mais uniquement lu. Quoi qu'il en soit chaque plugin doit fournir une fonction **register** lui permettant d'inscrire ses fonctionalités au sein de la factory globale.

Bibliographie

[LD] Liste documentaire, SETG-LD-MAJA-010-CS

[SDM] Software Developer Manual, SETG-MD-MAJA-010-CS

[MU] Manuel utilisateur (anglais) de Maja, SETG-MU-MAJA-010-CS

[PSD] Interfaces des produits Sentinel2, S2-PDGS-TAS-DI-PSD

[PSC] Interfaces des produits Muscate, PSC-SL-411-0032

[DA01] Spécification technique du besoin, SETG-STB-MAJA-0636-CNES

[DA02] Spécification of MACCS-ATCOR joint algorithms for level 2A product and of Venus level 3 product, MAJA-TN-WP1-008-CNES

[DA06] Technical specifications, Venus products specifications, VE-ST-GSSM-39-CNES

[DA07] Technical notes, Venus ground segment interfaces, VE-NT-GSSM-196-CNES

[DA100] Document de conception globale des chaines L2 et L3 Venus, VE-CG-GSSM-010-CS