Formation à l'Orfeo ToolBox: Solutions des Travaux Pratiques

OTB Team

2017

Contents

1 Avant-propos

La plupart des solutions proposées dans ce guide des solutions sont écrites en utilisant l'interface ligne de commande des applications de l'Orfeo ToolBox. Néanmoins, le lecteur pourra aisément reporter ces paramètres dans l'interface graphique si besoin.

2 Généralités

2.1 Utiliser Monteverdi et QGIS

Les manipulations demandées sont démontrées par le formateur à la fin de l'exercice.

2.2 Le mécanisme des applications Orfeo ToolBox

2.2.1 Message 1

Pour faire apparaître ce premier message, il suffit d'observer qu'une image Pléiades est codée sur 12 bits, qu'il ne devrait donc pas y avoir de valeurs de pixel supérieures à $2^{12} - 1 = 4095$. Nous allons donc utiliser l'application **BandMath** pour seuiller les pixels au dessus de cette valeur :

```
$ otbcli_BandMath -il image1.tif           \
                  -out decoded1.tif uint8  \
                  -exp "im1b1>4095?255:0"
```

Le texte encodé apparaît alors en blanc sur fond noir.

2.2.2 Message 2

Pour décoder ce deuxième message nous allons calculer un gradient en utilisant l'application **EdgeExtraction**:

```
$ otbcli_EdgeExtraction -in image2.tif    \
                        -filter gradient    \
                        -out decoded2.tif
```

2.2.3 Message 3

Pour décoder le troisième message, il suffit de calculer un indice de végétation de type NDVI à l'aide de l'application **RadiometricIndices**:

```
$ otbcli_RadiometricIndices -in image3.tif      \
                            -channels.red 1       \
                            -channels.nir 4       \
                            -list Vegetation:NDVI \
                            -out decoded3.tif
```

Alternativement, il est également possible de calculer le NDVI avec l'application **BandMath**:

```
$ otbcli_BandMath -il image3.tif                    \
                  -out decoded3.tif                 \
                  -exp "(im1b4-im1b1)/(im1b4+im1b1)"
```

2.2.4 Message 4

Afin de faire apparaître le quatrième message, nous allons isoler les 2 bits de poids faible en utilisant l'application **BandMath**:

```
$ otbcli_BandMath -il image4.tif              \
                  -out decoded4.tif           \
                  -exp "im1b1-4*rint(im1b1/4)"
```

L'expression $4 * rint(im1b1/4)$ ne contient aucun des 2 bits de poids faible, la différence avec l'image codée révèle donc le message.

2.2.5 Message 5

Pour révéler le cinquième message, nous allons réaliser une analyse en composantes principales à l'aide de l'application **DimensionalityReduction** et en extraire la dernière bande à l'aide de l'application **ExtractROI**, où se concentre le bruit:

```
$ otbcli_DimensionalityReduction -in image5.tif  \
                                 -out pca6.tif   \
                                 -method pca
$ otbcli_ExtractROI -in pca6.tif                 \
                    -out decoded6.tif            \
                    -cl Channel4
```

2.2.6 Message 6

Pour révéler le sixième message, nous allons nous appuyer sur la propriété d'idempotence. Si le message a été encodé à l'aide d'une transformation idempotente, alors $f(message) = message$, et par conséquent $f(message) - message = 0$, tandis qu'ailleurs dans l'image, on observera $f(image)$.

```
$ otbcli_GrayScaleMorphologicalOperation -in image6.tif             \
                                         -out ouverture6.tif        \
                                         -structype.ball.xradius 1  \
                                         -structype.ball.yradius 1  \
                                         -filter opening
$ otbcli_BandMath -il image6.tif ouverture6.tif                     \
                  -out decoded6.tif                                 \
                  -exp "(im2b1-im1b1)"
```

2.3 Les mécanismes internes de l'Orfeo ToolBox

2.3.1 Encodage des images

L'utilisation de **gdalinfo** pour l'image *image1.tif* nous donne:

```
$ $ gdalinfo image1.tif
Driver: GTiff/GeoTIFF
Files: image1.tif
Size is 2000, 2000
Coordinate System is `'
Origin = (5400.000000000000000,4300.000000000000000)
Pixel Size = (1.000000000000000,1.000000000000000)
Image Structure Metadata:
INTERLEAVE=PIXEL
Corner Coordinates:
Upper Left  (    5400.000,    4300.000)
Lower Left  (    5400.000,    6300.000)
Upper Right (    7400.000,    4300.000)
Lower Right (    7400.000,    6300.000)
Center      (    6400.000,    5300.000)
Band 1 Block=2000x1 Type=Float32, ColorInterp=Gray
Band 2 Block=2000x1 Type=Float32, ColorInterp=Undefined
Band 3 Block=2000x1 Type=Float32, ColorInterp=Undefined
Band 4 Block=2000x1 Type=Float32, ColorInterp=Undefined
```

Les pixels sont donc encodés en nombres flottants de 32 bits. En analysant les valeurs de l'image dans **monteverdi**, on constate que les valeurs de pixels sont entières et comprises entre 100 et 1600 environ. L'encodage en flottants de 32 bits est donc inutilement coûteux.

L'appel à l'application **Convert** permet de convertir le type de pixel encodé:

```
$ otbcli_Convert -in image1.tif -out image1_uint16.tif uint16
```

Nous pouvons maintenant comparer la taille des images, et constater que l'image ainsi générée occupe seulement la moitié de la place par rapport à l'image d'origine.

```
$ du -h image1.tif
62M image1.tif

$ du -h image1_uint16.tif
31M image1_uint16.tif
```

L'utilisation de l'application **CompareImages** nous montre par ailleurs que le contenu des deux images est identique.

```
$ otbcli_CompareImages -ref.in image1.tif -meas.in image1_uint16.tif
2016 Mar 08 13:59:24  :  Application.logger  (INFO) Using whole reference image
                         since the ROI contains no pixels or is not specified
2016 Mar 08 13:59:24  :  Application.logger  (DEBUG) Region of interest used
                         for comparison : ImageRegion (0x7ffcb6a6d930)
Dimension: 2
Index: [0, 0]
Size: [2000, 2000]

2016 Mar 08 13:59:24  :  Application.logger  (INFO) reference image channel 1
                         is compared with measured image channel 1
2016 Mar 08 13:59:24  :  Application.logger  (INFO) MSE: 0
2016 Mar 08 13:59:24  :  Application.logger  (INFO) MAE: 0
2016 Mar 08 13:59:24  :  Application.logger  (INFO) PSNR: 0
Output parameters value:
mse: 0
mae: 0
psnr: 0
```

Pour calculer le NDVI, on utilise la commande suivante :

```
$ otbcli_RadiometricIndices -in image1.tif
                            -out image1_ndvi.tif uint16
                            -channels.red 1
                            -channels.green 2
                            -channels.blue 3 -channels.nir 4
                            -list Vegetation:NDVI
```

Si l'on ouvre l'image ainsi générée dans **monteverdi**, on constate que l'image vaut 0 en tout point: l'encodage de la sortie ne convient pas. Il faudrait utiliser un type flottant (comme celui par défaut par exemple).

2.3.2 Les fichiers .geom

Le fichier geom contient les informations nécessaires aux opérations de corrections géométriques et radiométriques de l'image.

2.3.3 Les noms de fichiers étendus

Les options de lecture L'utilisation du paramètre de nom de fichier étendu *skipgeom* permet d'ignorer les informations contenue dans le fichier *geom*. On constate que la taille du pixel au sol est erronée et que les informations relatives à la date d'acquisition et au capteur ont notamment disparu.

Le paramètre de nom de fichier étendu *geom* permet d'attacher un fichier *geom* à une image existante. C'est notamment utile pour réaliser des traitements géométriques ou radiométriques pour une image quelconque. Par défaut, l'Orfeo ToolBox (en fait OSSIM) cherche un fichier *geom* portant le même nom que l'image.

Les options d'écriture La ligne de commande suivante permet de réaliser l'opération demandée:

```
$ otbcli_Convert -in image1.tif
  -out "image1_comp.tif?&gdal:co:NBITS=12&gdal:co:COMPRESS=LZW"  uint16
```

La taille de l'image ainsi créée est :

```
$ du -h image1_comp.tif
23M image1_comp.tif
```

On gagne donc 8 Mo par rapport à l'image encodée sur 16 bits non signés. Par ailleurs l'appel à l'application **CompareImages** permet de constater que les images sont toujours de contenu identique.

Le paramètre *box* s'utilise de la manière suivante :

```
$ otbcli_Convert -in image1.tif
                 -out "image1_comp.tif?&box=1000:1000:100:100"  uint16
```

Après exécution de cette commande, l'image de sortie correspond à un extrait de la sortie totale, commençant à l'index $(1000, 1000)$ et de taille 100x100 pixels. Cette option peut être utile pour pré visualiser le résultat d'un traitement avant de traiter l'image entière.

2.3.4 La gestion du streaming

L'appel à l'application **LocalStatiticsExtraction** se fait comme suit:

```
$ otbcli_LocalStatisticExtraction -in image1.tif -out image1_ls.tif
                                  -radius 9
```

On constate que le calcul s'effectue en plusieurs phases (charge des processeurs), entrecoupées de phases d'écriture sur le disque. Par défaut, c'est l'Orfeo ToolBox qui détermine le découpage optimal.

Pour désactiver complètement le streaming, il suffit d'utiliser les options de noms de fichier étendus suivantes:

```
$ otbcli_LocalStatisticExtraction -in image1.tif *
        -out "image1_ls.tif?&streaming:type=none" -radius 9
```

On peut constater dans ce cas que le calcul s'effectue en une seule fois, suivi d'une seule phase d'écriture sur le disque.

```
$ otbcli_LocalStatisticExtraction -in image1.tif
  -out "image1_ls.tif?&streaming:type=stripped \
  &streaming:sizemode=nbsplits&streaming:sizevalue=1000"
  -radius 9
```

Cette fois-ci, on observe de multiples phases de calcul suivies de phases d'écriture. Le temps de calcul peut être quasiment deux fois plus long, car un découpage trop important est sous optimal.

2.3.5 Le multi-threading

Voici comment fixer le nombre de threads à 1 :

```
$ export ITK_GLOBAL_DEFAULT_NUMBER_OF_THREADS=1
$ otbcli_LocalStatisticExtraction -in image1.tif
                                  -out "image1_ls.tif?&streaming:type=none"
                                  -radius 9
```

Dans ce cas, le temps de calcul est beaucoup plus important. On peut également constater qu'augmenter le nombre de threads au delà des capacités de la machine (nombre de coeurs du processeur) ne permet pas d'améliorer les temps de calcul.

3 Imagerie THR optique, des pré-traitements au SIG

3.1 Pré-traitements de l'imagerie THR optique

3.1.1 Corrections atmosphériques

Calcul réflectance TOA:

```
$ otbcli_OpticalCalibration \
-in phr_xs_osr_mipy.tif \
-out phr_xs_osr_mipy_toa.tif uint16 \
-level toa \
-milli 1

$ otbcli_OpticalCalibration \
-in phr_pan_osr_mipy.tif \
-out phr_pan_osr_mipy_toa.tif uint16 \
-level toa \
-milli 1
```

Calcul réflectance TOC:

```
$ otbcli_OpticalCalibration
-in phr_xs_osr_mipy.tif \
-out phr_xs_osr_mipy_toc.tif uint16 \
-level toc \
-milli 1

$ otbcli_OpticalCalibration
-in phr_pan_osr_mipy.tif \
-out phr_pan_osr_mipy_toc.tif uint16 \
-level toc \
-milli 1
```

On peut utiliser le module **BandMathX** pour calculer la différence entre les 2 images multispectrales:

```
$ otbcli_BandMathX
-il phr_xs_osr_mipy_toa.tif phr_xs_osr_mipy_toc.tif \
-out diff_xs_toa_toc.tif int16 \
-exp "im1-im2"
```

Pour l'image panchromatique:

```
$ otbcli_BandMath
-il phr_pan_osr_mipy_toa.tif phr_pan_osr_mipy_toc.tif \
-out diff_pan_toa_toc.tif int16 \
-exp "im1b1-im2b1"
```

On constate que la bande bleue est la plus sensible aux effets atmosphériques. En effet c'est pour la bande bleue que l'influence de la diffusion moléculaire sur le signal est maximale (effet en λ^{-4}).

3.1.2 Fusion P+XS

```
$ otbcli_BundleToPerfectSensor \
-inp phr_pan_osr_mipy_toa.tif \
-inxs phr_xs_osr_mipy_toa.tif \
-mode phr \
-out phr_pxs_osr_mipy.tif uint16
```

3.1.3 Ortho-rectification

1. Orthorectification sans DEM:

   ```
   $ otbcli_OrthoRectification \
   -io.in phr_pxs_osr_mipy.tif \
   -io.out phr_orthopxs_osr_mipy.tif uint16
   ```

2. Orthorectification avec DEM et geoid:

   ```
   $ otbcli_OrthoRectification \
   -io.in phr_pxs_osr_mipy.tif \
   -io.out phr_orthopxs_osr_mipy.tif uint16 \
   -elev.dem SRTM/ \
   -elev.geoid Geoid/egm96.grd
   ```

3. La projection par défaut est UTM. Sur l'extrait Pléiades la zone UTM est 32 Nord.

4. Orthorectification en WGS84 et en Lambert 93:

   ```
   $ otbcli_OrthoRectification \
   -io.in phr_pxs_osr_mipy.tif \
   -io.out phr_orthopxs_osr_mipy.tif uint16 \
   -elev.dem SRTM/ \
   -elev.geoid Geoid/egm96.grd \
   -map epsg -map.epsg.code 4326

   $ otbcli_OrthoRectification \
   -io.in phr_pxs_osr_mipy.tif \
   -io.out phr_orthopxs_osr_mipy.tif uint16 \
   -elev.dem SRTM/ \
   -elev.geoid Geoid/egm96.grd \
   -map lambert93
   ```

3.2 Segmentation et export vers un SIG

3.2.1 Lissage de l'image par l'algorithme MeanShift

L'étape de lissage se réalise de la manière suivante:

```
$ otbcli_MeanShiftSmoothing -in phr_orthopxs_osr_mipy_xt.tif
  -fout meanshift.tif
  -foutpos meanshift_pos.tif
  -ranger 25
  -spatialr 3
  -maxiter 10 -modesearch 0
```

Le paramètre *spatialr* correspond au rayon spatial du lissage. Une valeur plus élevée provoquera un lissage plus fort, mais également un temps de calcul supérieur.

Le paramètre *ranger* correspond au rayon spectral du lissage, c'est à dire dans quelle mesure les pixels à l'intérieur du rayon spatial et de radiométrie similaire seront moyennés. Une valeur plus élevée augmentera l'effet de lissage.

L'image *foutpos* n'a pas de sens visuellement, et sera utilisée pour la suite de l'exercice.

3.2.2 Segmentation

L'étape de segmentation se réalise de la manière suivante:

```
$ otbcli_LSMSSegmentation -in meanshift.tif
                          -inpos meanshift_pos.tif
                          -out init_seg.tif uint32
                          -ranger 10
                          -spatialr 2
```

L'image de segmentation ainsi créée est difficilement interprétable. On peut la coloriser de la manière suivante:

```
$ otbcli_ColorMapping -in init_seg.tif
                      -method optimal
                      -out init_seg_cm.tif uint8
```

Cet algorithme de colorisation analyse les segments adjacents pour maximiser leur contraste lors de la colorisation.

L'image segmentée colorisée peut être analysée, et l'on constate qu'il y a une grande quantité de petites régions qui ne correspondent à aucun objet précis de la scène. A noter que ces petites régions peuvent soit être filtrées en utilisant le paramètre *minsize* de l'application **LSMSSegmentation**, soit être traitées dans l'étape suivante.

3.2.3 Traitement des petites régions

Le traitement des petites régions s'effectue comme suit:

```
$ otbcli_LSMSSmallRegionsMerging -in meanshift.tif
                                 -inseg init_seg.tif
                                 -out final_seg.tif uint32
                                 -minsize 100
```

On peut ensuite coloriser à nouveau le résultat de la manière suivante:

```
$ otbcli_ColorMapping -in final_seg.tif
                      -method optimal
                      -out final_seg_cm.tif uint8
```

En comparant les deux segmentations, on peut constater que les régions de taille inférieur au paramètre spécifié ont été fusionnées avec les régions voisines les plus pertinentes.

3.2.4 Vectorisation

Pour commencer, on calcule l'indice NDVI pour l'image initiale:

```
$ otbcli_RadiometricIndices -in phr_orthopxs_osr_mipy_xt.tif
                            -out phr_ndvi.tif
                            -list Vegetation:NDVI
                            -channels.blue 3
                            -channels.red 1
                            -channels.green 2
                            -channels.nir 4
```

Ensuite, on peut concaténer l'image initiale avec l'image de NDVI:

```
$ otbcli_ConcatenateImages -il phr_orthopxs_osr_mipy_xt.tif phr_ndvi.tif
                           -out phr_radio_ndvi.tif
```

Enfin, on réalise la vectorisation:

```
$ otbcli_LSMSVectorization -in phr_radio_ndvi.tif
                           -inseg final_seg.tif -out segmentation.shp
```

En ouvrant la table des attributs dans QGIS, on constate qu'on peut accéder pour chaque polygone à la moyenne et à la variance de chaque bande de l'image (incluant le NDVI).

3.2.5 Filtrage polygones dans QGIS

Pour sélectionner tous les segments qui ne sont pas des ombres à l'aide de l'outil de sélection par expression, on peut utiliser l'expression suivante:

```
meanB0 > 140 or meanB1 > 140 or meanB2 > 140 or meanB3 > 140
```

Ensuite, en utilisant la calculatrice de champ, on peut créer un nouveau champ (virtuel) en réel, appelé *compac* en utilisant la formule suivante:

```
sqrt(area($geometry)/perimeter($geometry)
```

Enfin, pour sélectionner les petits objets compacts dont la valeur moyenne de NDVI est forte, on peut utiliser l'expression suivante dans l'outil de sélection par expression:

```
compac > 0.1 and nbPixels < 500 and meanB4 > 0.2
```

4 Classification supervisée pour les séries multi-temporelles

Dans la correction suivante, la variable d'environnement `$DATA` correspond au répertoire contenant les données du TP `TODO`.

4.1 Réaliser un apprentissage mono-date

L'apprentissage mono-date se réalise avec la commande suivante :

```
$ otbcli_TrainImagesClassifier -io.il $DATA/images/20160607_T31TCJ_ROI_20m.tif \
                               -io.vd $DATA/references/training/training.shp   \
                               -io.out model.rf                                \
                               -sample.vfn CODE -classifier rf                 \
                               -classifier.rf.nbtrees 50 -classifier.rf.max 20 \
                               -classifier.rf.cat 13
```

Cette première exécution donne les résultats suivants :

```
Confusion matrix (rows = reference labels, columns = produced labels):
       [10]  [31]  [32]  [34]  [36]  [41]  [42 ] [43]  [44]  [51] [211] [221] [222]
[ 10]   374     3     0    26     1     6    19    11     2     0    13    10    13
[ 31]     0   436     5     7    14     0     0     0     0     0     7     8     1
[ 32]     3    16   420     6    15     0     0     0     0     0    12     3     3
[ 34]    30    16    21   268    41     1    13     2     1     0    27    18    40
[ 36]    10     6    13    31   336     0     7     0     0     0    42    13    20
[ 41]     5     0     0     0     0   388    49    21    13     0     0     1     1
[ 42]    31     1     3    10     3    44   288    36    22     0     0     5    35
[ 43]    18     0     2    11     1    37    59   227   114     1     0     7     1
[ 44]     7     0     3     3     2     5    10    71   371     0     0     5     1
[ 51]     0     0     6     0     0     0     0     0     1   470     0     1     0
[211]    19     7    13    41    64     0     3     0     0     0   266    14    51
[221]    18    18     8    13    23     1    11     4     3     0    38   332     9
[222]    22     0     1    30    12     0    14     0     0     0    16     4   379

[...]

Global performance, Kappa index: 0.710774
```

Ces performances sont cependant très optimistes, car les échantillons utilisés pour les estimer proviennent des mêmes polygones. Pour obtenir une évaluation plus réaliste des performances, il faut utiliser un jeu de validation différent :

```
$ otbcli_TrainImagesClassifier -io.il $DATA/images/20160607_T31TCJ_ROI_20m.tif \
                               -io.vd $DATA/references/training/training.shp   \
                               -io.valid $DATA/references/testing/testing.shp  \
                               -io.out model.rf                                \
                               -sample.vfn CODE -classifier rf                 \
                               -classifier.rf.nbtrees 50 -classifier.rf.max 20 \
                               -classifier.rf.cat 13
```

Ce qui donne les résultats suivants :

```
Confusion matrix (rows = reference labels, columns = produced labels):
       [10]  [31]  [32]  [34]  [36]  [41]  [42]  [43]  [44]  [51] [211] [221] [222]
[ 10]   795     6     6    47     9     4    13    18     0     0    22    10    23
[ 31]     1   777    38    14    42     0     0     0     0     1    59    21     0
[ 32]     1    34   865     3    12     2    14     1     1     0     2    15     3
[ 34]    50   273   120    72    51     0     8     0     0    49   105   157    68
[ 36]    23    27    45   186   336     0     1     1     0     0   215    79    40
[ 41]     4     0     1     1     0   665   176    53    49     0     1     1     2
[ 42]    20     1     3    11     2    98   652    75    43     0     5     8    35
[ 43]    21     0     1    19     5    44   207   464   146     1     8    17    20
[ 44]     7     0     1     3     0    13    23   240   662     0     0     4     0
[ 51]     0     0     1     0     0     0     0     3     1   945     0     3     0
[211]    81    21    17    81   112     1    16     4     0     0   507    40    73
[221]    46    51    22    43    42     0    18    10     2     2   107   541    69
[222]    70     0     0    68    23     0    71     1     0     0    45    11   664

[...]

Global performance, Kappa index: 0.611403
```

Si l'on désactive l'option `cleanup` en ajoutant le paramètre `-cleanup 0`, l'application n'efface pas les sorties intermédiaires générées.

Les fichiers XML suivants contiennent les statistiques de nombre d'échantillons disponibles par classe pour le jeu d'apprentissage et celui de validation.

- `model.rf_statsTrain_1.xml`
- `model.rf_statsValid_1.xml`

Les fichiers Shapefile suivants contiennent les échantillons utilisés pour l'apprentissage et pour la validation:

- `model.rf_samplesTrain_1.shp`
- `model.rf_samplesValid_1.shp`

Ces fichiers contiennent des points correspondant aux échantillons sélectionnés dans les polygones d'apprentissage. Chaque point contient un ensemble de primitives qui correspond aux radiométries mesurées à cet endroit dans l'image. Ces deux fichiers peuvent être affichés dans Qgis.

4.2 Identifier la date la plus performante

La commande suivante permet de réaliser l'apprentissage pour chaque date.

```
$ for f in $DATA/images/*.tif; do echo $f;            \
      otbcli_TrainImagesClassifier -io.il $f          \
      -io.vd $DATA/references/training/training.shp   \
      -io.valid $DATA/references/testing/testing.shp  \
      -sample.vfn CODE -classifier rf                 \
      -classifier.rf.nbtrees 50 -classifier.rf.max 20 \
      -classifier.rf.cat 13 -io.out model.rf   | grep Kappa; done
```

Les coefficients Kappa par date généré par cette commande sont les suivants :

Date	Kappa
2016-06-07	0.609741
2016-07-07	0.615163
2016-08-06	0.593739
2016-09-05	0.614463
2016-10-05	0.622246

On constate que ce coefficient ne varie pas beaucoup, mais que la date du `2016-10-05` obtient des performances légèrement meilleures.

4.3 Réaliser la classification et produire une carte en couleur

Pour réaliser la classification, on prend le fichier `model.rf` appris sur la date `2016-10-05`, et on utilise la commande suivante :

```
$ otbcli_ImageClassifier -in $DATA/images/20161005_T31TCJ_ROI_20m.tif \
                         -out classif_20161005.tif uint8              \
                         -model model.rf
```

L'image `classif_20161005.tif` contient pour chaque pixel le code de la classe qui lui a été attribué. Afin de faciliter la lisibilité de l'image, on peut transformer celle-ci en attribuant une couleur différente à chaque classe en utilisant l'application de **ColorMapping**:

```
$ otbcli_ColorMapping -in classif_20161005.tif            \
                      -out classif_20161005_rgb.tif uint8 \
                      -method custom -method.custom.lut   \
                      $DATA/support/color_map.txt
```

Une autre manière de visualiser l'image `classif_20161005.tif` est de l'ouvrir dans QGis et d'utiliser le fichier de style fournit dans `support/classif.qml`.

4.4 Évaluer la performance globale

Pour évaluer les performances sur l'ensemble de la donnée de validation, on utilise l'application **ComputeConfusionMatrix**. Cette application complète l'évaluation réalisée lors de l'apprentissage, et permet d'évaluer la performance d'une carte de classification qui a éventuellement été retraitée. Attention à ne pas utiliser en entrée la carte colorisée créée à l'étape précédente, qui n'est utile qu'à des fins de visualisation et de publication.

```
$ otbcli_ComputeConfusionMatrix -in classif_20161005.tif -ref vector  \
                  -ref.vector.in $DATA/references/testing/testing.shp \
                                 -out confusion_20161005.csv          \
                                 -ref.vector.field CODE
```

La performance est évaluée en utilisant l'ensemble des données disponibles dans le jeu de validation. Voici le résultat :

```
Confusion matrix (rows = reference labels, columns = produced labels):
       [  10] [  31] [  32] [  34] [  36] [  41] [  42] [  43] [  44] [  51] [ 211] [ 221] [ 222]
[  10] 113219    219   2240  10349   4090   2654   2469   1029    233    202   7387   2571   3453
[  31]      8  12282    265     66    346      0      5      1      0      0    174    192     27
[  32]      1     21   1143      5     27      0      2      0      0      0      3      7     10
[  34]    158     47     73   1469    146      9     68     25     12      0    187     11     70
[  36]     11      8      2      8    889      0      4      0      0      0     25      1     11
[  41]     45      0      4     34      7   4637    674    287    135      1      9      9     66
[  42]    800     14    107    675    355   5084  33947   3642   2684     13    468    490   3358
[  43]    816      4     97    417    130   2222   5399  18726   9404     66    137    171    857
[  44]     12      0      7      7      5     54    105    561   2807      6      0     13     27
[  51]    187     26     92     10     73      1     18    249    257  24330      4    300      4
[ 211]    367     73     55    882   1143      9     58      1      0      0   6755    126    301
[ 221]    244    337    372     79    338      2    197     49     16     32    174  10400    398
[ 222]     72      2     66     40    115      9    195      1      0      0     98     52   3474

Precision of class [10] vs all: 0.976531
Recall of class [10] vs all: 0.754215
F-score of class [10] vs all: 0.851095

[...]

Kappa index: 0.659139
```

On peut constater deux choses:

- Tout d'abord, la performance globale est légèrement meilleure que celle évaluée lors de l'apprentissage. Cela vient du fait que dans l'étape d'apprentissage, le même nombre d'échantillon est utilisé pour chaque classe, tandis que lors du calcul ci-dessus, l'ensemble des échantillons disponibles est utilisé. Certaines classes plus représentée et bien reconnue, comme la classe 51 (eau), tirent donc les performances globales vers le haut.
- Ensuite, la classe des cultures annuelles présente une assez forte confusion avec l'ensemble des autres classes. Elle présente un rappel de 0.75, et une précision de 0.97. Cela signifie que si 97% des éléments identifiés comme cultures annuels par le classifieur appartiennent effectivement à cette classe, 25% des éléments de cette classe dans la référence ont été mal classés. Nous allons voir par la suite que l'ajout d'une information multi-temporelle permet d'améliorer cette performance.

4.5 Régulariser et mesurer le gain de performance

Pour réaliser une régularisation par vote majoritaire, on utilise la commande suivante:

```
$ otbcli_ClassificationMapRegularization -ip.radius 1 -ip.suvbool 0  \
                                         -io.in classif_20161005.tif \
                                         -io.out classif_20161005_reg.tif uint8
```

Si l'on évalue à nouveau les performances, on obtient :

```
$ otbcli_ComputeConfusionMatrix -in classif_20161005_reg.tif -ref vector  \
                    -ref.vector.in $DATA/references/testing/testing.shp    \
                                   -out confusion_20161005_reg.csv         \
                                   -ref.vector.field CODE
```

```
Kappa index: 0.709103
```

La régularisation améliore donc significativement les performances. Ceci s'explique par la régularité de la donnée de référence, dont on se rapproche avec ce type de traitement.

4.6 Réaliser une classification multi-date et mesurer le gain de performance

Rejouons les différentes étapes avec l'ensemble des dates:

Tout d'abord, l'apprentissage :

```
$ otbcli_TrainImagesClassifier -io.il $DATA/images/all.vrt                      \
                               -io.vd $DATA/references/training/training.shp    \
                               -io.valid $DATA/references/testing/testing.shp   \
                               -io.out model_all.rf                             \
                               -sample.vfn CODE -classifier rf                  \
                               -classifier.rf.nbtrees 50 -classifier.rf.max 20 \
                               -classifier.rf.cat 13
```

Ensuite, la classification :

```
$ otbcli_ImageClassifier -in $DATA/images/all.vrt    \
                         -out classif_all.tif uint8  \
                         -model model.rf
```

Puis la régularisation :

```
$ otbcli_ClassificationMapRegularization -ip.radius 1 -ip.suvbool 0  \
                                         -io.in classif_all.tif \
                                         -io.out classif_all_reg.tif uint8
```

Enfin, l'évaluation des performances globales:

```
$ otbcli_ComputeConfusionMatrix -in classif_all_reg.tif -ref vector       \
                    -ref.vector.in $DATA/references/testing/testing.shp    \
                                   -out confusion_all_reg.csv              \
                                   -ref.vector.field CODE
```

```
Confusion matrix (rows = reference labels, columns = produced labels):
       [  10] [  31] [  32] [  34] [  36] [  41] [  42] [  43] [  44] [  51] [ 211] [ 221] [ 222]
[  10] 140681     13     68   1893    790    280    494    545     27    108   3569    383   1264
[  31]     19  12732     87     77    200      3      8      0      0      0    104    131      5
[  32]      1      6   1211      0      1      0      0      0      0      0      0      0      0
[  34]      0     32     23   2131     34      0     12     13      0      0     19      5      6
[  36]      0      0      0      4    937      0      4      0      0      0      8      0      6
[  41]      2      0      1      0      0   5369    330    111     87      1      0      7      0
[  42]    148     10     75    465     84   3166  41818   2943   1850      2    316    362    398
[  43]    143      6     67    528     57   1545   5556  21781   8265     35    166    195    102
[  44]     11      0     14     13      0     34     60    354   3108      0      2      8      0
[  51]      7     18     53      0     12      0      3     37     24  25319      2     76      0
[ 211]    213     41     17    444    491      8     45      1      0      0   8326    104     80
[ 221]    187     87     66    123    159      0    109     83      3     14    208  11374    225
[ 222]     29      0      2     15     41      4     42      0      0      0     43     11   3937

Precision of class [10] vs all: 0.994627
Recall of class [10] vs all: 0.937155
F-score of class [10] vs all: 0.965036

[...]

Kappa index: 0.828411
```


L'ajout de l'information multi-temporelle dans la classification a permis d'améliorer significativement les performances. On peut constater notamment que le rappel de la classe cultures annuelles a augmenté jusqu'à 93%, ce qui signifie que désormais 93% des éléments de cette classe présents dans la vérité terrain sont correctement identifiés par le classifieur. Cette amélioration était attendue car les classes de cultures présentent une dynamique temporelle distinctive, par rapport à d'autres classes.

On peut enfin générer la carte de classification colorisée finale :

```
$ otbcli_ColorMapping -in classif_all_reg.tif            \
                      -out classif_all_reg_rgb.tif uint8 \
                      -method custom -method.custom.lut  \
                      $DATA/support/color_map.txt
```

5 Traitements SAR pour l'imagerie Sentinel 1

5.1 Introduction au traitements des images RSO

Introduction à l'imagerie RSO

1. Les 2 extraits correspondent respectivement à la combinaison polarimétrique HH (transmission et réception horizontales) et HV (transmission horizontale et réception verticale).
2. Ces bandes correspondent respectivement à la partie réelle et partie imaginaire du signal radar.
3. On peut utiliser l'application **BandMath** pour réaliser le calcul de l'image d'intensité:

   Pour HH:

   ```
    $ otbcli_BandMath \
   -il s1_hh.tif \
   -out intensity_hh.tif int32 \
   -exp "im1b1*im1b1+im1b2*im1b2"
   ```

   Pour HV:

   ```
   $ otbcli_BandMath \
   -il s1_hv.tif \
   -out intensity_hv.tif int32 \
   -exp "im1b1*im1b1+im1b2*im1b2"
   ```

Calibration radiométrique

1. **SARCalibration**
2. Pour Sentinel-1 les coefficients de calibration sont lus automatiquement dans les métadonnées du produit:

   ```
   $ otbcli_SARCalibration \
   -in s1_hh.tif \
   -out s1_hh_gamma0.tif \
   -lut gamma
   ```

   Pour l'extrait de l'image en polarisation HV:

   ```
   $ otbcli_SARCalibration \
   -in s1_hv.tif \
   -out s1_hv_gamma0.tif \
   -lut gamma
   ```

3. Attention aux pixels <= 0 dans l'expression du log!

```
$ otbcli_BandMath \
-in s1_hh_gamma0.tif \
-out s1_hh_gamma0_db.tif \
-exp "im1b1>0?10*log10(im1b1):0"
```

Et pour HV:

```
$ otbcli_BandMath \
-in s1_hv_gamma0.tif \
-out s1_hv_gamma0_db.tif \
-exp "im1b1>0?10*log10(im1b1):0"
```

Corrections géométriques On utilise une grille de déformation à 10 mètres de résolution pour accélérer le traitement.

1. Orthorectification sans DEM:

```
$ otbcli_OrthoRectification \
-io.in s1_hh_gamma0.tif \
-opt.gridspacing 10 \
-io.out s1_hh_gamma0_ortho.tif uint16
```

2. Orthorectification avec DEM et geoid:

```
$ otbcli_OrthoRectification \
-io.in s1_hh_gamma0.tif \
-io.out s1_hh_gamma0_ortho.tif uint16 \
-opt.gridspacing 10 \
-elev.dem SRTM/ \
-elev.geoid Geoid/egm96.grd
```

3. La projection par défaut est UTM. Sur l'extrait Sentinel-1 la zone UTM est 32 Nord.

Filtrage du speckle

1. Les méthodes disponibles sont: Lee, Frost, Kuan et Gamma MAP. Quelque soit la méthode utilisée on note une amélioration majeure de la qualité de l'image filtrée qui permet d'identifier des structures difficilement visibles dans l'image d'intensité originale.

2. Réduction du speckle avec l'algorithme de Frost:

```
$ otbcli_Despeckle \
-in intensity_hh.tif \
-out intensity_hh_speckle.tif \
-filter frost \
-filter.frost.rad 3
```

L'augmentation du rayon a pour effet d'augmenter le lissage de l'image filtrée. Cela permet d'améliorer la qualité des images dans les zones homogènes mais entraîne également la perte d'informations et de détails sur des petites structures avec beaucoup de contraste.

3. L'histogramme des images filtrées tend à devenir gaussien (en cloche) et va progressivement différer de la distribution Gamma de l'image originale (la loi Gamma se caractérise par une distribution en cloche asymétrique avec une longue queue à droite) .
4. L'augmentation du paramètre *deramp* diminue la décroissance de l'atténuation exponentielle et à donc tendance à prendre plus en compte les pixels éloignés du pixel central ce qui augmente l'effet de lissage sur l'image filtrée.

Polarimétrie

1. Calcul de la différence HH-HV:

```
$ otbcli_BandMath \
-il intensity_hh_speckle.tif intensity_hv_speckle.tif \
-out hh-hv_speckle.tif \
-exp "im1b1-2*im2b1"
```

2. On effectue ensuite la concaténation entre les polarisations croisées et la différence des 2:

```
$ otbcli_ConcatenateImages \
-il intensity_hh_speckle.tif \
intensity_hv_speckle.tif hh-hv_speckle.tif \
-out intensity_compo.tif
```

1. Attention aux pixels <= 0 dans l'expression du log!

```
$ otbcli_BandMath \
-in intensity_compo.tif \
-out intensity_compo_db.tif \
-exp "im1b1>0?10*log10(im1b1):0"
```

2. Commentaires:
   - Layover: correspond à un effet géométrique réponse similaire HH et HV
   - Les variabilités traduisent aussi des différences de type et de niveau de croissance des végétations et d'humidité du sol
   - Zone de végétation (forêt): vert/jaune
   - HV moins sensible à la rugosité
   - Zone en eau: réponse radar faible (HH)
3. Analyse de la composition colorée:
   - 2 lignes électriques parallèles autour des coordonnées (230,3700)
   - Coin réflecteur aux coordonnées image (3620,2925). C'est peut être un coin réflecteur fixe positionné pour la validation géométrique de Sentinel-1 (cette zone fait partie des zones de validation de la mission). Je n'ai pas trouvé d'information permettant de vérifier cette hypothèse.
   - Plots métalliques pour amarrer les bateaux

6 Développer avec l'OTB

6.1 Tutoriel pour futur développeur OTB

Les solutions de tous les exercices sont fournis aux élèves sous la forme d'une archive contenant les corrigés de tous les fichiers sources.