Exploit Pleiades PHR data with the ORFEO ToolBox Updated for OTB-4.2

Manuel Grizonnet (CNES), Julien Michel (CNES), Jordi Inglada (CNES)

November 14, 2014

Contents

1 How to build

- Use org-mode
- You can generate pdf using org-babel
- Solutions can be exclude from the tutorial using the `exclude_tags` export option

2 Foreword

2.1 About the data

The images used during these exercise are extracts from Pleiades demonstration products, made available for evaluation purpose. To get the full products please refer to this website. Products used are:

- `Pléiades Primary Product - Bundle`
- `Pléiades Ortho Product - Pan-sharpened`
- `Pleiades TRISTEREO Bundle PRIMARY`

Those images are located on Melbourne, Australia and are covered by a CNES copyright.

Exercises based on these Pleiades data include a set of command-line to generate the needed extracts.

Other data needed for some exercises, as well as solution scripts can be found in the data package.

2.2 About the software

To perform the exercises, you will need to have the following software installed:

- **Orfeo ToolBox** 4.2 or later, including applications
- **Monteverdi2** 0.8 or later

- **QGIS** 2.0 or later

  For **Orfeo ToolBox** and **Monteverdi** installation, you can refer to the installation from the Orfeo ToolBox Cookbook.

  For **QGIS** installation, please refer to **QGIS** documentation, which can be found on the project website.

- **Monteverdi** 1.22 or later

3 Exercises

3.1 Monteverdi2 and OTB applications

3.1.1 Description

Abstract This exercise will get you familiar with the use of **Monteverdi2** and **OTB applications**.

Monteverdi2 is the successor of the Monteverdi application which allows image visualization and also to build processing based on OTB.

The development started in 2012 with regular release and an iterative development process whit short loop which include user feedbacks.

What's new (different from Monteverdi first version) in Monteverdi2?

- GUI revamp: the software interface using the Qt toolkit
- Modern viewer (navigation in resolutions, multiple color dynamic modes, fast OpenGL rendering...)
- Session manager : database manager to store datasets and display parameters
- Processing : interactive access to OTB application tools

This tutorial is an adaptation of the one dedicated to Monteverdi and focus on Monteverdi2 main features.

Figure 1: Monteverdi2 main window

Data In this exercise we will use extract from PMS and MS products. If you need to generate the data used in this exercise from the original products, you can use the following command lines.

```
$ otbcli_ExtractROI \
 -in IMG_PHR1A_PMS-N_201202250025599_ORT_PRG_FC_5855-001_R1C1.JP2 \
 -out phr_pxs_melbourne.tif uint16 -startx 18432 -starty 4096 -sizex 4096 -sizey 4096

$ otbcli_ExtractROI \
  -in IMG_PHR1A_MS_201202250025599_SEN_PRG_FC_5847-002_R1C1.JP2 \
  -out phr_xs_melbourne.tif uint16 -startx 4096 -starty 2048 -sizex 4096 -sizey 4096
```

Pre-requisites

- Basic knowledge of remote sensing and image processing,

Achievements

- Visualize data in **Monteverdi2**,
- Basic processing in **Monteverdi2** (rendering methods, get dataset and pixel information),
- Basic processing with **OTB applications** in graphical mode

3.1.2 Steps

Monteverdi2: data opening and visualisation In this part of the exercise, you will use the following data: `phr_pxs_melbourne.tif` and `phr_pan_melbourne.tif`

1. Run **Monteverdi2**: open a terminal and run the following command:

   ```
   $ monteverdi2
   ```

2. Open the image (use /File/Import image...)
3. After opening Monteverdi2, the application runs through the importation process before opening the application. In your opinion what are the main processing done by the application?
4. Navigate into the image:
   (a) Change the full resolution displayed area using the QL window
   (b) Change the zoom level,
   (c) How to navigate through resolution in Monteverdi2?
   (d) Point the mouse cursor on the navigation view in a region with shadows and right click. What happens?
5. Using the dataset properties widget
   (a) What is the image size?
   (b) What is the meaning of the information "Block size"?
6. Using the Pixel description widget
   (a) What are the information displayed about the current pixel under mouse pointer ?
   (b) How to know in Monteverdi2 if the image displayed is georeferenced?
7. Using the Color video setup widget

(a) Change set-up to visualize the 4th band,

(b) Change set-up to visualize in false color mode (screen red: near infra-red channel, screen green: red channel, screen blue: green channel).

(c) Change set-up to come back to natural colors

8. Using the preferences widget

   (a) Monteverdi2 allows to specify a directory containing DEM information. Can you give one example of RS processing where a DEM is needed?

Tips and Recommandations:

- Starting with Monteverdi2 0.7 you need to press *Ctrl* and use the mouse wheel at the same time to navigate through image resolution,
- Pleiades bands order is red channel, green channel, blue channel, near infra-red channel.

Monteverdi2: basic rendering functions and histogram visualization In this part of the exercise, you will use the following data:

`phr_xs_melbourne.tif`

1. What is the default rendering function apply in Monteverdi2
2. What are the 2 different ways to change the rendering threshold?
3. What is the behaviour when you move the cursor of the gamma correction widget?
4. What is the purpose of the "No data" setup?

OTB applications: Graphical and command-line mode

1. Run the following command:

   ```
   $ otbcli_OrthoRectification
   ```

   then

   ```
   $ otbgui_OrthoRectification
   ```

   and then searh for the Orthorectification application from the applications What do you observe ?

2. How many **OTB applications** are currently available ?
3. What is the purpose of the **ram** parameter in the Orthorectification application?
4. How can you get help and documentation about applications ?
5. Do you know other graphical software which allow to access to OTB applications?

OTB applications in Monteverdi2: Basic processing In this part of the exercise, you will use the following data:

`phr_xs_melbourne.tif`

1. Open the image in **Monteverdi2**.
2. Find the *BandMath* application in the menu. Import the image in this module. What kind of processing is offered ?
3. Using this module, compute the NDVI of the image:

$$NDVI = \frac{NIR - RED}{NIR + RED} \tag{1}$$

4. What is the path of the output image file?
5. Visualize the output in Monteverdi2
6. Using this application, build a mask of pixels whose Digital Number (DN) in the NIR channel is lower than 150. Switch between the input image and the mask.
7. Using this application, build a mask of pixels whose DN is upper than 1000 in all spectral bands.
8. Using the *Images concatenation* app, build a composite RGB image with the mask of high values in the red channel, the mask of low NIR values in the blue channel and the NDVI in the green channel.
9. Using the *Color Mapping* app, build a composite RGB image of the NDVI that allows for better image interpretation.

Tips and Recommandations:

- NDVI values are within -1 and 1, but the range can be much more narrow.
- Import dataset as input of applications by drag and drop inside the applications widget

Homework

1. Is it possible to load or visualize images directly from command-line using **Monteverdi2** ?
2. What is the geoid?
3. Is there another way to compute radiometric indices like NDVI with the **OTB Applications** ?
4. Is there a way to save applications parameters values (input/output images, parameters) in a file for future reuse?
5. Learn about the *Python* access to **OTB Applications** and write a python script performing the same steps as in section 3.1.2

3.1.3 Solutions

Monteverdi2: data opening and visualisation

1. Item 3 Monteverdi2 will try to generate external overviews image to speed-up navigation in resolution. It stores also the histogram of the Quick Look (QL) to allow histogram visualization and rendering settings. It will also save information of the session (channel composition, viewer position, rendering settings...).

   All those parameters are store internally in a SQLite database

2. Item 4 Use *Ctlr* and the mouse wheel to navigate through resolution with Monteverdi2.

   Right click triggered a "magnifying glass" for local contrast enhancement. The apply formula is:

   $$255 \times (\frac{x - min}{max - min})^{gamma}$$

   The exponent is used for gamma correction and *min* and *max* are the image X percent minimum and maximum values.

3. Item 5 Image size is available in the *Dataset properties* widget (4096,4096)

   The *Block size* corresponds to the way that bitmapped data is organized in memory. For instance, for the TIFF format, images can be dividing an image into rectangular tiles rather than horizontal strips which can have benefit on very large high-resolution images.

   If you want to learn more about writing large images and the way that TIFF format store data, see Writing large images.

   OTB is using GDAL to read/write images and writing TIFF format with gdal produce by default stripped data (1 line all columns). In this case

   This default behaviour can be set in OTB using *Extended Filename* mechanism, see Extended Filename.

4. Item 6 The status bar display the index position and the radiometric values of the pixel under the mouse pointer. Other information are available in the *Pixel Description* widget.

   The *Pixel Description* widget provides information about geographic position of the pixel under the mouse pointer.

   If the image is georeferenced, the geographic position is marked as **Exact**. If no geo-information are available, OTB will try to estimate geographic position using sensor models (based on OSSIM library). In this case the position is marked as **Sensor model**.

5. Item 8 A digital elevation model is a digital model or 3D representation of a terrain's surface used for instance for rectification of satellite imagery, surface analysis...

Monteverdi: basic rendering functions and histogram visualization

1. Item 1 Linear rescale using the histogram 2% minimum and maximum values

2. Item 2 You can set lower and upper quantile in percentage or set min/max values use for threshold.

   This two variable can be linked/unlinked using the padlock icon. Unlinking the two controls allow to specify Min/Max values outside the range of the computed histogram.

3. Item 3

   Gamma correction is a process that can compensate from the production of the image, the fact that the acquisition process get DN proportional to the illumination which can leads to information *too dark*. The inverse gamma curve applied to images will therefore clarify and spread the tonal range to produce a more or less linear visual picture.

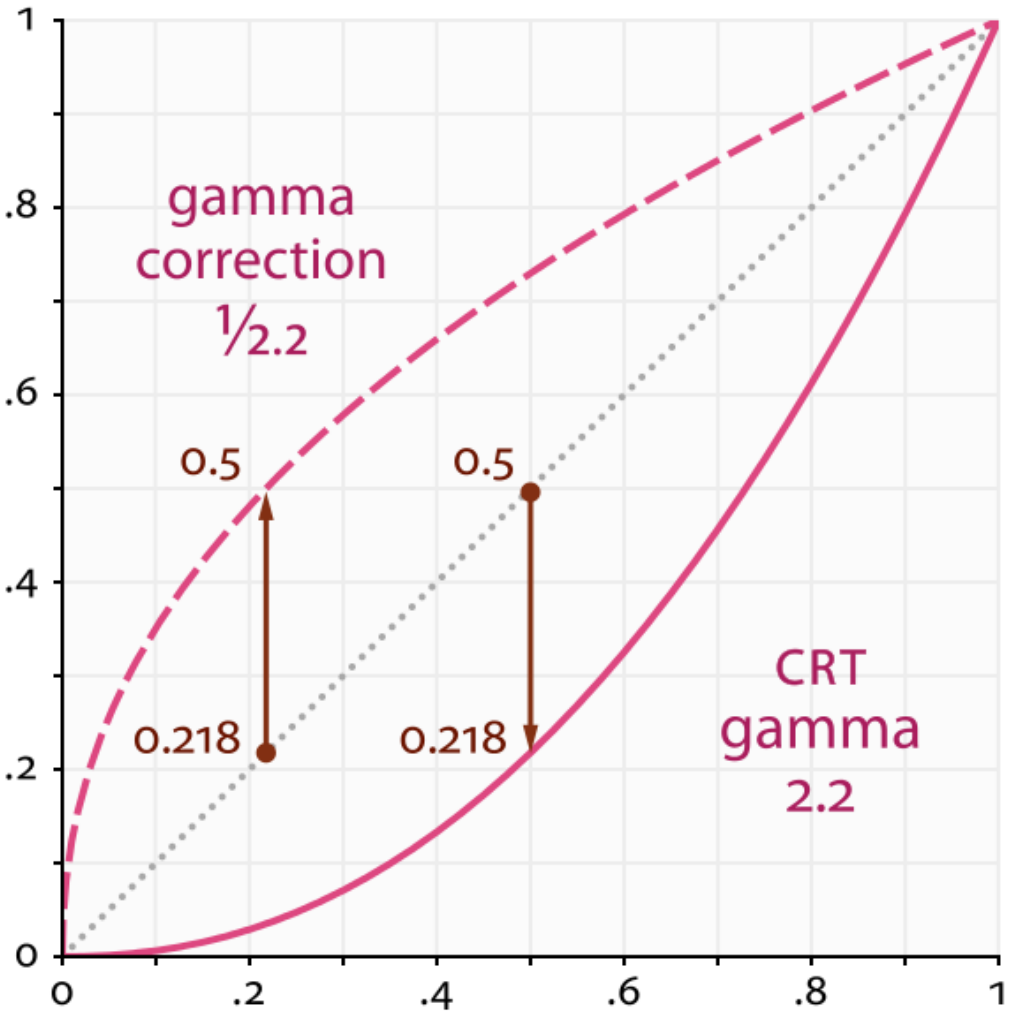


Figure 2: Example of CRT gamma correction (Wikipedia)

4. Item 4

   No data corresponds to pixel values not taken into account into the rendering methods. It could be useful for instance with ortho products which can be rotated and contain a large number of *black* pixels (radiometry equal to zero)

OTB applications: Graphical and command-line mode

1. Item 1

The first command runs the command-line version of the **Orthorectification** application, the second one runs the graphical version.

2. Item 2

   The number of OTB applications vary depending of the OTB version. All applications are listed in the OTB Cookbook. You can also try the following command in your *bin* directory:

   ```
   ls ~/local/bin/otbcli_* | wc -l
   ```

3. Item 3

   It allows to set the maximum amount of RAM available for processing. As the writing task is time consuming, it is better to write large pieces of data, which can be achieved by increasing this parameter (pay attention to your system capabilities).

   Note that this value can be underestimate.

4. Item 4

   There are several ways to get help and documentation:

   - Running the command-line version of the application displays a short description of the parameters, and also gives a link to the documentation on the OTB website,
   - Running the graphical version of the application shows a *Documentation* tab where extensive documentation of parameters can be found.
   - Last, the complete applications documentation can be found in the Orfeo ToolBox Cookbook.

5. Item 4 OTB applications are also available through the **Processing** module in QGIS.

OTB applications in Monteverdi2: Basic processing

1. Item 1 Output images of OTB applications in Monteverdi2 are stored by default in a *Result* directory. This directory is located by default in the cache directory. The output file is filled by default with an automatic path using Universally unique identifier.

2. Item 2 Here is the set of commands to reproduce the processing from section `OTB applications in Monteverdi2: Basic processing`.

   You can get parameters values that should be setted in the graphical mode of OTB applications integrated in Monteverdi2

   First, we compute the NDVI with the **BandMath** application:

   ```
   $ otbcli_BandMath -il phr_xs_melbourne.tif
     -out ndvi.tif float -exp "(im1b4-im1b1)/(im1b4+im1b1)"
   ```

   Then, we compute the mask of pixels whose DN in the NIR channel is lower than 150:

   ```
   $ otbcli_BandMath -il phr_xs_melbourne.tif
     -out lownir.tif uint8 -exp "if(im1b4<150,255,0)"
   ```


Next, we compute the mask of pixels whose DN is upper than 1000 in all spectral bands:

```
$ otbcli_BandMath -il phr_xs_melbourne.tif
  -out high.tif uint8
  -exp "if(min(im1b1,im1b2,im1b3,im1b4)>1000,255,0)"
```

Please note that for masks using a *uint8* data type is enough, while for NDVI a floating point data type is needed.

Now, we can concatenate all outputs in a single map with the **ConcatenateImages** application:

```
$ otbcli_ConcatenateImages -il high.tif ndvi.tif lownir.tif
  -out map1.tif float
```

Finally, we can create a color-mapping of the NDVI using the **ColorMapping** application:

```
$ otbcli_ColorMapping -in ndvi.tif -out map2.png uint8
  -method continuous -method.continuous.min -0.2
  -method.continuous.max 0.7 -method.continuous.lut jet
```

Homework

1. Item 1 Not yet
2. Item 2 The geoid is the shape that the surface of the oceans would take under the influence of Earth's gravitation (source Wikipedia).

   In case of tasks involving sensor to ground and ground to sensor coordinate transforms (like during ortho-rectification),these transforms need to find the intersection between the line of sight of the sensor and the earth geoid. If a simple spheroid is used as the earth model, potentially high localisation errors can be made in areas where elevation is high or perturbed.
3. Item 3 In **OTB Applications**, there is the **RadiometricVegetationIndices** application that allows to compute several indices including the NDVI.
4. Item 4 You can save parameters values for all applications in an XML file using the *outxml* parameter.

   Therefore, you can relaunch the application with the exact same parameter using the *inxml* parameter.Note that these parameters values can also be overwrited in the application.
5. Item 5

   Please refer to this chapter of the **Cookbook** to learn more about the *Python* interface.

3.2 Pre-processing : geometry and radiometry corrections

3.2.1 Description

Abstract This exercise will get you familiar with the geometric and radiometric corrections using **OTB applications**.

Data If you need to generate the data used in this exercise from the original products (Bundle MS), you can use the following command lines.

```
$ otbcli_ExtractROI \
  -in IMG_PHR1A_MS_201202250025599_SEN_PRG_FC_5847-002_R1C1.JP2 \
  -out phr_xs_melbourne.tif uint16 -startx 4096 -starty 2048 -sizex 4096 -sizey 4096
```

Pre-requisites

- Basic knowledge of remote sensing and image processing,
- Basic knowledge of command-line invocation.

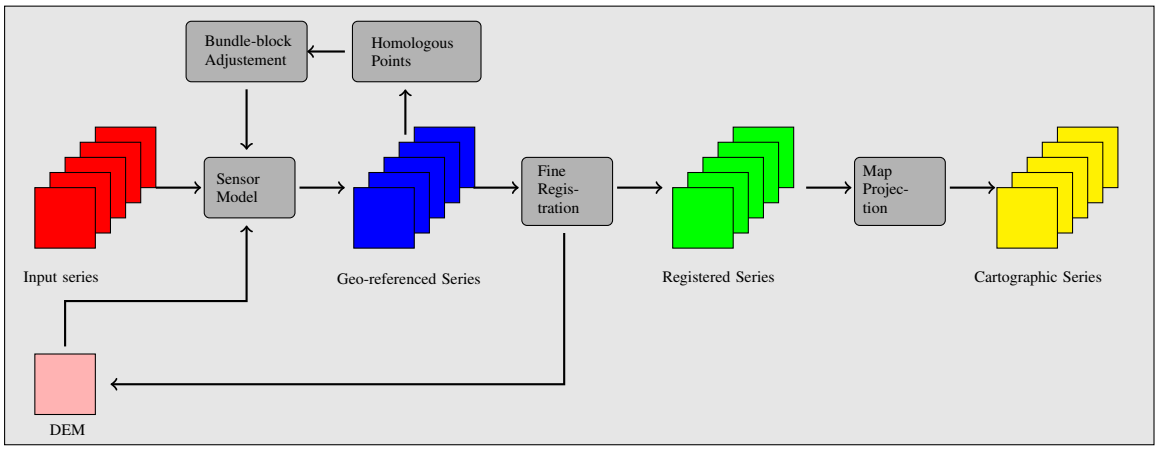
Achievements

- Orthorectification of remote sensing image using **OTB Applications**
- Optical calibration of remote sensing image using **OTB Applications**

3.2.2 Steps

In this part of the exercise, you will use the following data: `phr_xs_melbourne.tif`

Geometric corrections principle This operation allows to go from image index to ground coordinates.



1. Run the command-line and graphical version of the **Orthorectification** application
2. Which parameters allow to manage Digital Elevation Model informations (DEM) in the application? Which modes are available?
3. Open the image with Monteverdi, what is the estimated spacing of the input image?

Orthorectification

1. In which UTM zone, is located Melbourne?
2. Use Quantum GIS to determine the corresponding EPSG code for this projection.
3. Perform the orthorectification of the image and open it in Quantum GIS

Tips and Recommandations:

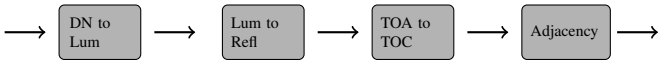
- Use the estimated spacing read in **Monteverdi** to set the output spacing parameters in the **Orthorectification** application
- Use an average elevation of 20 meters

Optical calibration This operation allows to go from Digital Number (DN) to reflectance (physical values)

It includes to convert Top Of Atmosphere to Top Of Canopy reflectance which aim is to compensate the atmospheric effects.

The optical calibration framework is OTB is fully adapted to the pipeline architecture.

The aim of this operation is to obtain physical values from images.



1. Use the **OpticalCalibration** application to compute Top Of Atmosphere reflectance.
2. Use the **OpticalCalibration** application to compute Top Of Canopy reflectance.
3. Compare the two images using Monteverdi or OTB applications.

Tips and Recommandations:

- Enable the '-milli' parameter which allow to save the output image in uint16. By default, reflectance image is saved in float values (between 0 and 1)

3.2.3 Solutions

Geometric corrections principle

1. Item 2

   The *elev* group of parameters allows to manage DEM informations

2. Item 3

   The estimated spacing in Monteverdi is *(1.95;2.03)*

Orthorectification

1. Item 1

   The UTM zone of Melbourne is 55. See here for example.

2. Item 2

   The EPSG code is 32755. This information is available in the Quantum GIS project properties window.

3. Item 3

   Here is the command-line to run the orthorectification :

```
$ otbcli_Orthorectification -io.in phr_xs_melbourne.tif
  -io.out solution/phr_xs_melbourne_ortho.tif uint16
  -map utm -map.utm.zone 55
  -outputs.spacingx 2. -outputs.spacingy -2.
  -interpolator bco
  -elev average -elev.average.value 20
```

Optical calibration

1. Item 1

   Here is the command-line to compute TOA reflectance :

```
$ otbcli_OpticalCalibration -io.in phr_xs_melbourne.tif
  -io.out solution/phr_xs_melbourne_toa.tif uint16
  -milli 1
```

2. Item 2

   Here is the command-line to compute TOC reflectance :

```
$ otbcli_OpticalCalibration -io.in phr_xs_melbourne.tif
  -io.out solution/phr_xs_melbourne_toc.tif uint16
  -milli 1 -level toc
```

3. Item 3

   You can use the **BandMath** module to compute the difference between the *TOA* and *TOC* images

3.3 Segmentation

3.3.1 Description

Abstract This exercise will get you familiar with the OTB **Segmentation** application. You will learn how to produce a raster segmentation output with different algorithms and how to scale up to larger input images by producing vector outputs.

Data If you need to generate the data used in this exercise from the original products (Ortho PMS), you can use the following command lines.

```
$ otbcli_ExtractROI \
 -in IMG_PHR1A_PMS-N_201202250025599_ORT_PRG_FC_5855-001_R1C1.JP2 \
 -out segmentation_small_xt_phr.tif uint16 -startx 11848 -starty 11426 -sizex 1024 -sizey 1024

$ otbcli_ExtractROI \
 -in IMG_PHR1A_PMS-N_201202250025599_ORT_PRG_FC_5855-001_R1C1.JP2 \
 -out segmentation_large_xt_phr.tif uint16 -startx 10240 -starty 10240 -sizex 4096 -sizey 4096
```

Pre-requisites

- Basic knowledge on **OTB applications** and **QGIS** usage
- Basic knowledge on image segmentation
- Basic knowledge on GIS vector file formats

Achievements

- Usage of the OTB **Segmentation** application,
- Segmentation of large raster and import the results in a GIS software.

3.3.2 Steps

Getting familiar with the Segmentation application

1. Run the command-line and graphical version of the application
2. Read the documentation. What are the three segmentation methods available ?
3. What are the two output modes ?

Simple segmentation in raster mode In this part of the exercise, you will use the following data: `segmentation_small_xt_phr.tif`

1. Run the **Segmentation** application in *raster* mode, using the connected components filter and a thresholding condition on the spectral distance
2. View the resulting segmentation in **Monteverdi**. What do you see ?
3. Use the **ColorMapping** application to enhance the rendering of the result:
   (a) Try the *optimal* method
   (b) Try the *image* method
4. Try different connected components conditions and see how they influence the results. You can try to change the distance threshold for instance, or look into the documentation for other keywords.

Tips and Recommandations:

- Use the **distance** keyword in the expression to denote spectral distance
- Pay attention to the output image type

More segmentation algorithms In this part of the exercise, you will use the following data: `segmentation_small_xt_phr.tif`

1. Run the **Segmentation** application in *raster* mode again, but this time use the Mean-Shift filter. Use the **ColorMapping** application to visualize the results.
   (a) Try the default parameters first
   (b) Try to change the parameters and see how it influences the results. The most important parameters are the spatial and the range radius.
2. Run the **Segmentation** application in *raster* mode again, but this time use the Watershed filter. Use the **ColorMapping** application to visualize the results.
   (a) Try the default parameters first
   (b) Try to change the parameters and see how it influences the results.
3. Compare the best results from the three algorithms. Keep the best segmentation result you had for Exercise 3.

Tips and Recommendations:

- There are two implementations of the Mean-Shift filter. Edison is the original implementation from the Mean-Shift paper authors.

Going big: the vector mode In this part of the exercise, you will use the following data: `segmentation_large_xt_phr.tif`

1. Run the **Segmentation** application in *raster* mode again, using the best parameters you had in previous section, on the large image. Look at computer resources. What happens ?
2. Run the **Segmentation** application again, this time in *vector* mode, and **disable the stitching option**. Look at computer resources. What happens ?
3. Open the result of the input image and the segmentation file in **QGis**. Tune **QGis** to allow for proper visualization (see Tips and Recommendation). What do you see ?
4. Run the **Segmentation** application again, this time in *vector* mode, and **enable the stitching mode**. Write the results to a different file and load it into the **QGis** project as well. What is the effect of the **stitch** option ?

Tips and Recommendations:

- Computer resources can be monitored by running `top` in another terminal
- Hit `Ctrl C` to interrupt the processing
- Use the *sqlite* file format to store vector outputs (`.sqlite` file extension)
- In **QGis**, one can import both raster and vector layers
- In **QGis**, one can tune raster layers rendering the following way:

  - Right-click on the layer, select *Properties*
  - Go to the *style* tab
  - Select *Use standard deviation*
  - In *Contrast enhancement*, select *Stretch to MinMax*

- In **QGis**, one can tune vector layers rendering the following way:
  - Right-click on the layer, select *Properties*
  - In the *style* tab, select *Change*
  - As *Symbol layer type*, select *Outline: Simple line*
  - You might change the color as well

- In **QGis**, you can save your project to a file and avoid having to reset those parameters

Homework

1. In *vector* mode, study the effect of the *tilesize*, *simplify* and *minsize* option.
2. Using the **Segmentation** application (and maybe other OTB applications), how can we segment everything but vegetation ?
3. Using the **Segmentation** application (and maybe other OTB applications), how can we deal with segmentation of high reflectance structures ?
4. Perform the same operation as in the "Vector mode" section using the LSMS framework (Large Scale Mean-Shift segmentation) which allows to perform exact segmentation at large scale without having to deal with tile effects.

3.3.3 Solutions

Getting familiar with the Segmentation application

1. Item 1

   To get the command-line help, run

   ```
   $ otbcli_Segmentation
   ```

   To Get the graphical version of the **Segmentation** application, run

   ```
   $ otbgui_Segmentation
   ```

2. Item 2

   There are three segmentation methods available in the application:

   - Mean-Shift (two different implementations)
   - Watershed (ITK implementation)
   - Connected-Components

3. Item 3

   There are two outputs available in the application:

   - The raster mode allows to segment a small image and produces a raster where each component of the segmentation is labeled with a unique integer,
   - The vector mode allows to segment larger images and produces a vector file where each segment of the segmentation is represented by a polygon.

Simple segmentation in raster mode

1. Item 1

   Here is the command-line to run, using a threshold of 30 on the spectral distance:

```
$ otbcli_Segmentation -in segmentation_small_xt_phr.tif
  -filter cc -filter.cc.expr "distance < 30"
  -mode raster -mode.raster.out first_cc.tif uint32
```

   Please note that we use `uint32` as the output type so as to be sure to have enough unique labels for the whole segmentation.

2. Item 2

   The segmentation result is difficult to visualize because neighboring segments are likely to be labeled with very close labels. One can notice the brightness gradient from top to bottom corresponding to globally increasing labels.

3. Item 3

   The following command-line allow to use the **ColorMapping** application in optimal mode:

```
$ otbcli_ColorMapping -in first_cc.tif
  -out first_cc_color_optimal.png uint8
  -method optimal
```

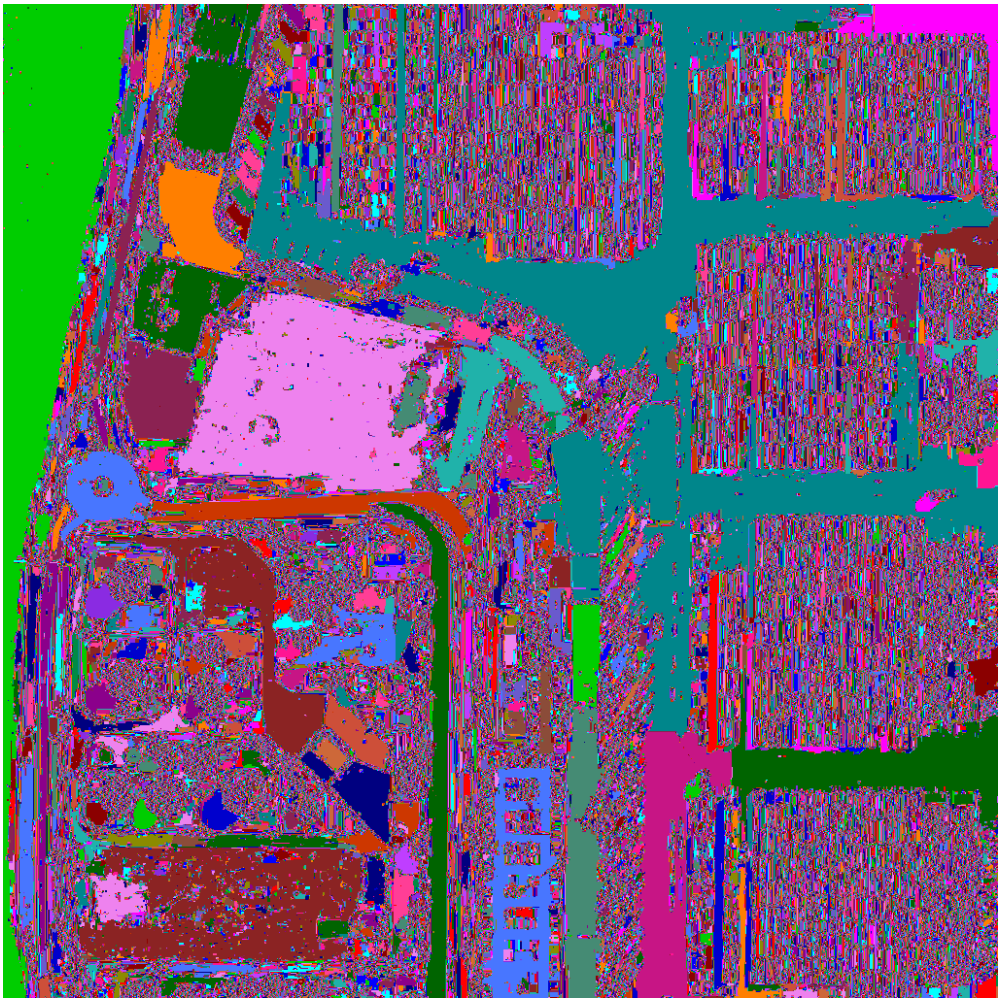
   The *optimal* color-mapping method allows to colorize each segment with a color maximizing contrast with the color of its neighbors. Please note that we use `uint8` as the output type because the **ColorMapping** application produces 8-bits data that can be directly viewed by any image viewer.

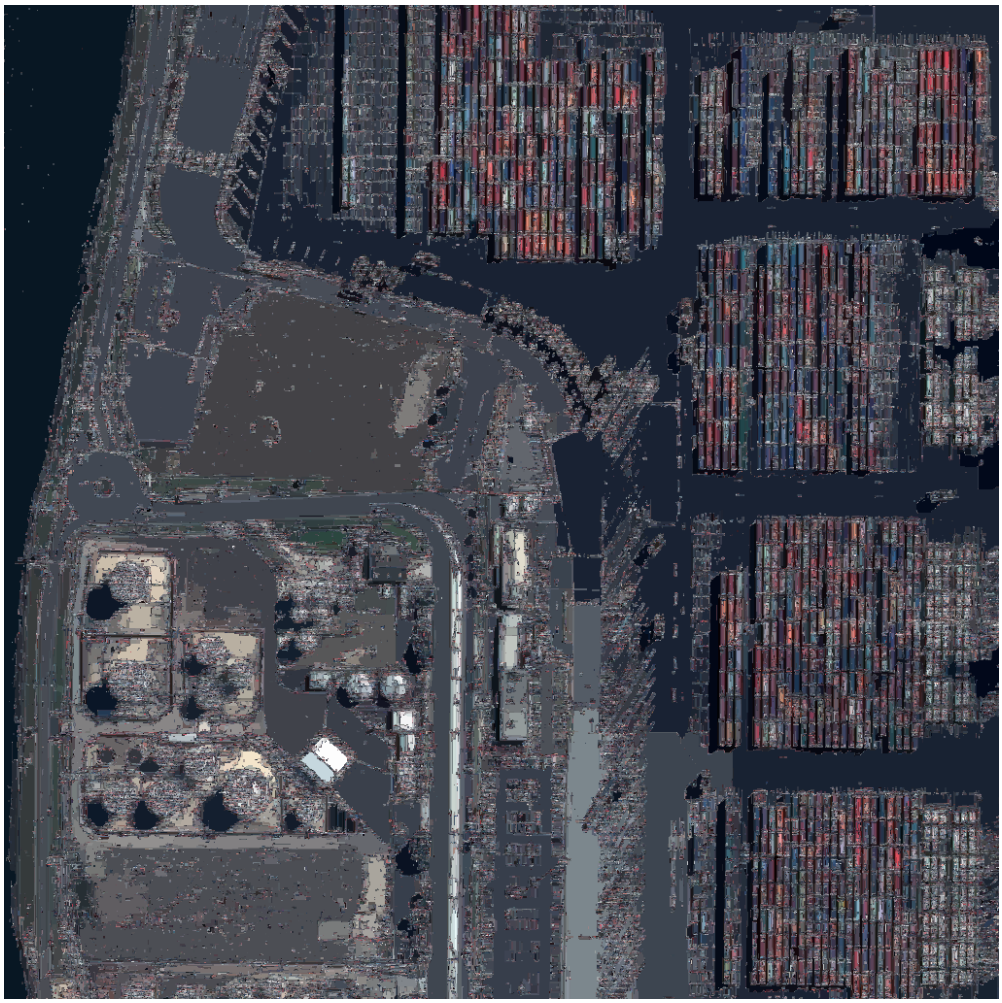
   Looking at the colorized image with the *optimal* look-up table, we can now see that the result is over-segmented.

```
$ otbcli_ColorMapping -in first_cc.tif
  -out first_cc_color_image.png uint8
  -method image -method.image.in segmentation_small_xt_phr.tif
```


The *image* color-mapping method allows to colorize each segment with its mean color in the original image, which gives a more realistic rendering. Note that since the results are over-segmented, the application will output a huge amount of text to the terminal.

Here are the results of the *optimal* (left) and *image* (right) methods:





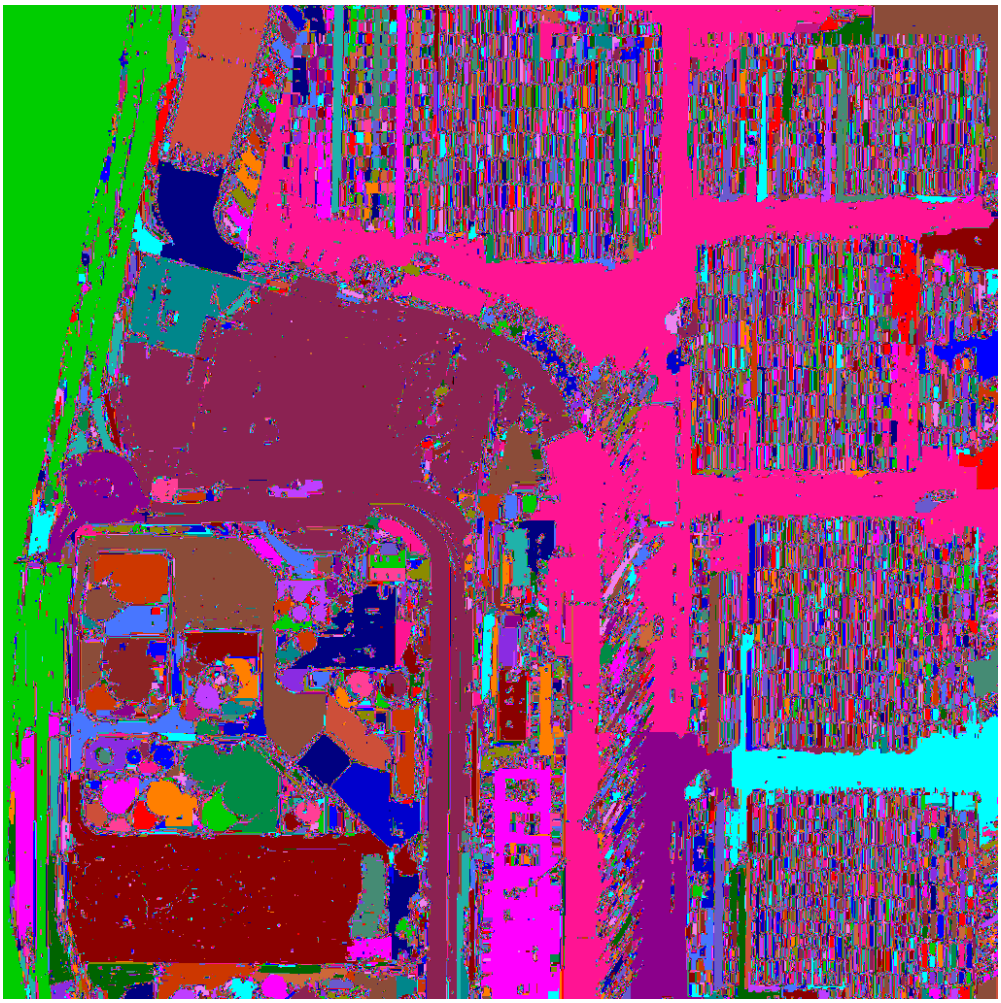
4. Item 4

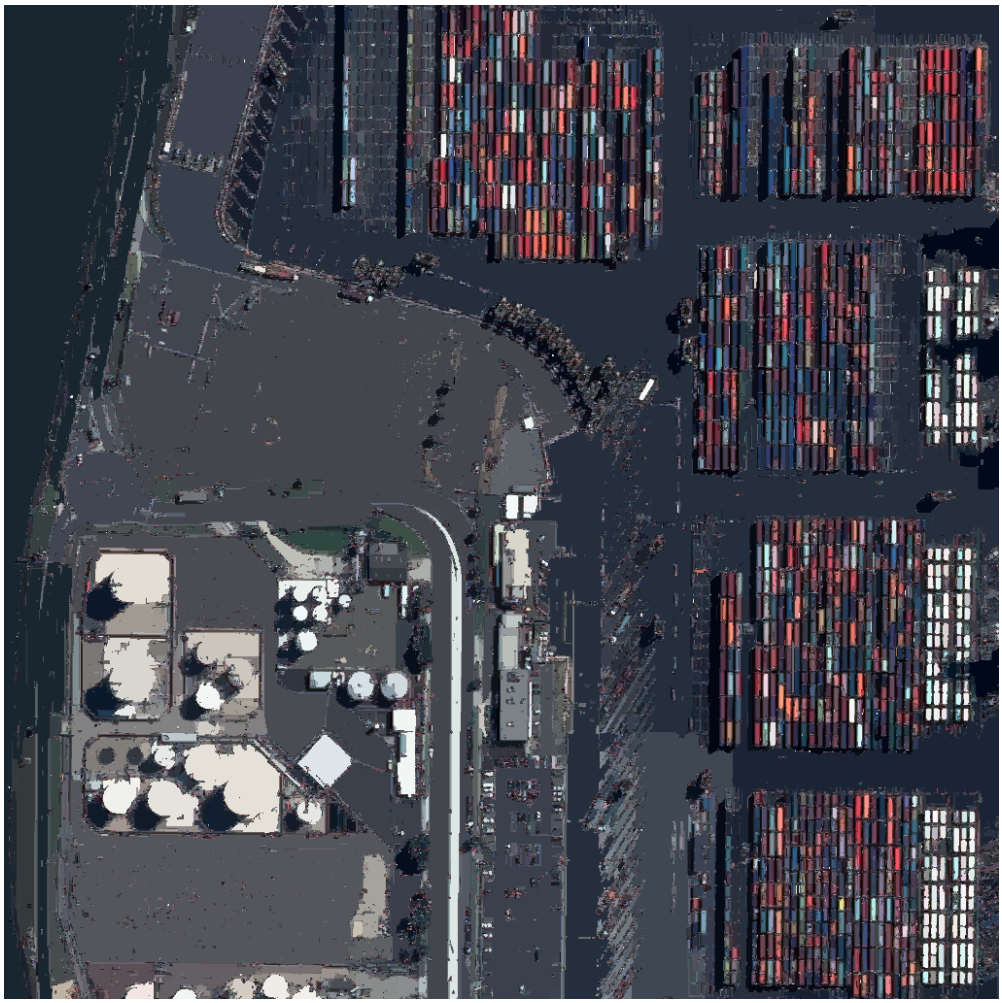
   Here is another example: the following command-line will segment together pixels that either:

   - Have a spectral distance lower than 30,
   - Have both an intensity value greater than 400 and a spectral distance lower than 50,
   - Have both an intensity value greater than 1000,
   - Have both a near infra-red value lower than 150.

```
$ otbcli_Segmentation -in segmentation_small_xt_phr.tif
  -filter cc -filter.cc.expr "distance<30
   or (intensity_p1>400 and intensity_p2 > 400 and distance<50)
   or(intensity_p1 >1000 and intensity_p2>1000
   or (p1b4 <150 and p2b4<150))"
  -mode raster -mode.raster.out second_cc.tif uint32
```

   Here are the color-mapping results:





More segmentation algorithms

1. Item 1

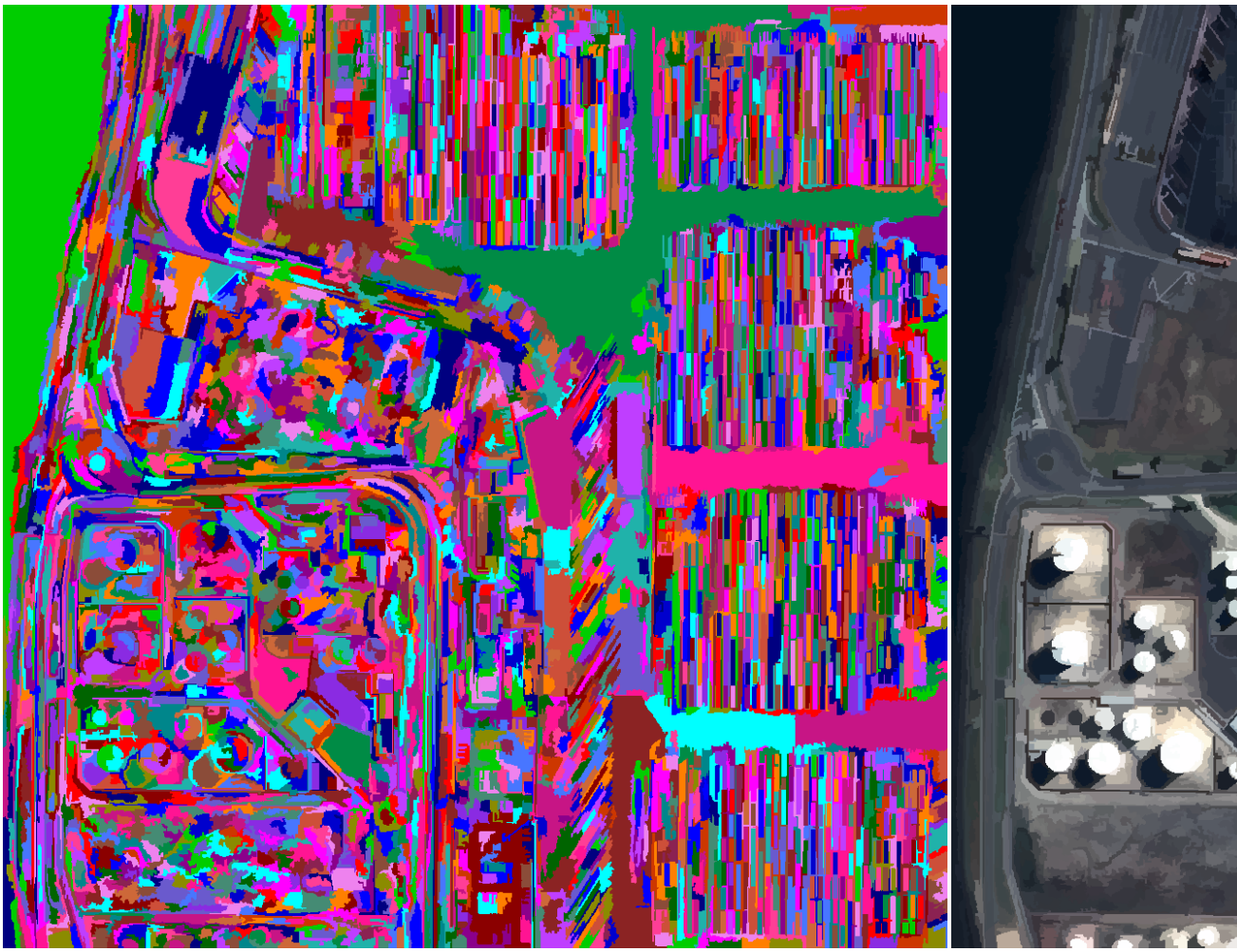
   Here is the command-line to run the application using the Mean-Shift filter, with default parameters:

   ```
   $ otbcli_Segmentation -in segmentation_small_xt_phr.tif
     -filter meanshift -mode raster
     -mode.raster.out meanshift.tif uint32
   ```

   A better result is obtained by using a spectral radius of 30:

   ```
   $ otbcli_Segmentation -in segmentation_small_xt_phr.tif
     -filter meanshift -filter.meanshift.ranger 30 -mode raster
     -mode.raster.out meanshift.tif uint32
   ```

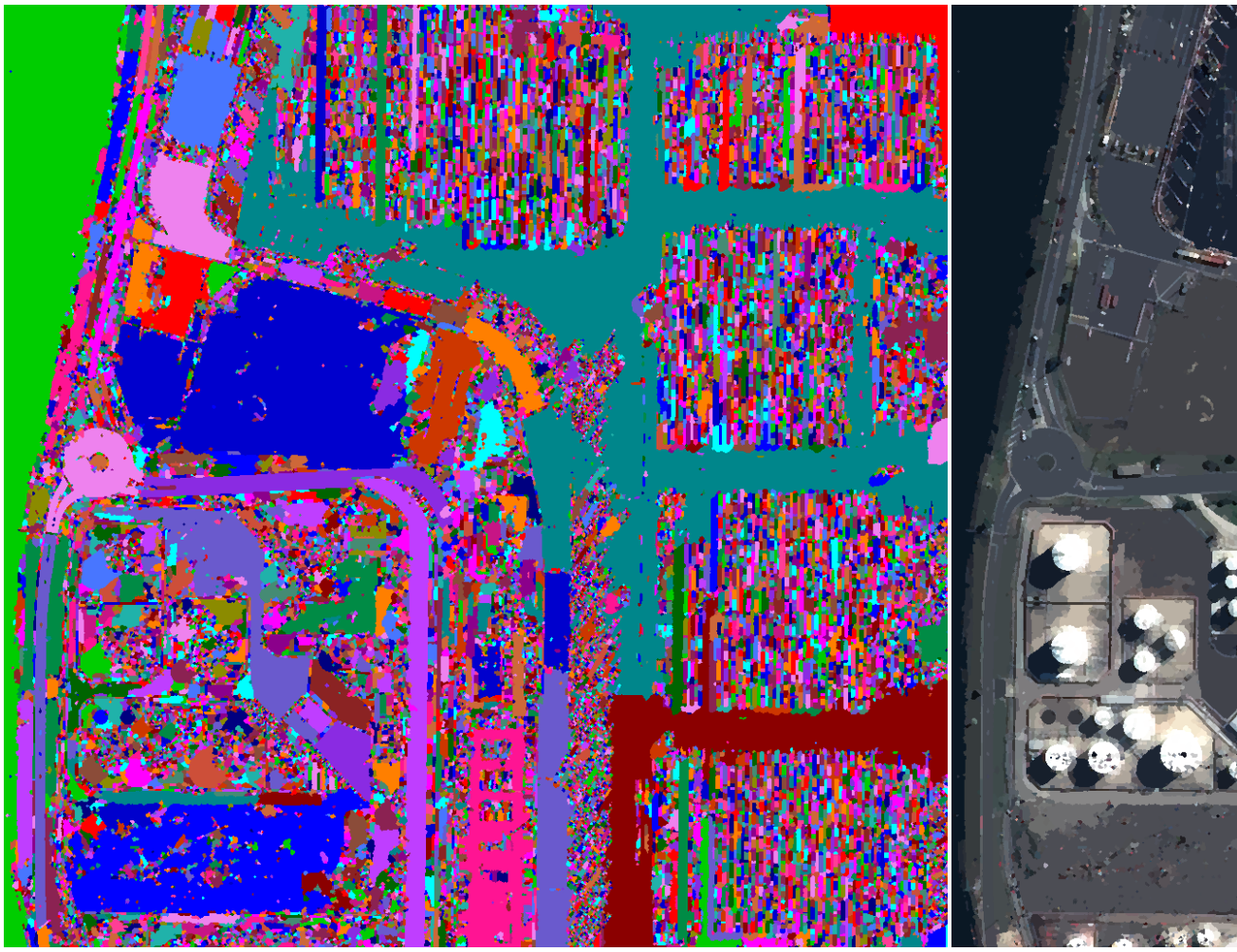
   Here are the results of this command:



Here is the command-line to run the application using the Watershed filter, with default parameters:

```
$ otbcli_Segmentation -in segmentation_small_xt_phr.tif
  -filter watershed -mode raster
  -mode.raster.out watershed.tif uint32
```

Here are the results of this command:



Going big: the vector mode

1. Item 1

   The following command will run the application on the larger image:

   ```
   $ otbcli_Segmentation -in segmentation_large_xt_phr.tif
     -filter meanshift -filter.meanshift.ranger 30 -mode raster
     -mode.raster.out meanshift.tif uint32
   ```

   Since the input image is quite large (8192 by 8192 pixels), it is likely that, depending on the available memory on the computer:

   - The application fails with a memory allocation error,
   - The application does not fails but starts to eat all the available memory.

2. Item 2

   The following command will run the application in *vector* mode, without the *stitch* option:

```
$ otbcli_Segmentation -in segmentation_large_xt_phr.tif
  -filter meanshift -filter.meanshift.ranger 30 -mode vector
  -mode.vector.out meanshift.sqlite -mode.vector.stitch 0
```

In vector mode, the memory consumption is stable because the segmentation on a per tile basis.

3. Item 3

   In **QGis** we can see the effect of this tile-based segmentation : tiles border are visible in the segmentation result. On can also see that the segmentation produces a large number of polygons.

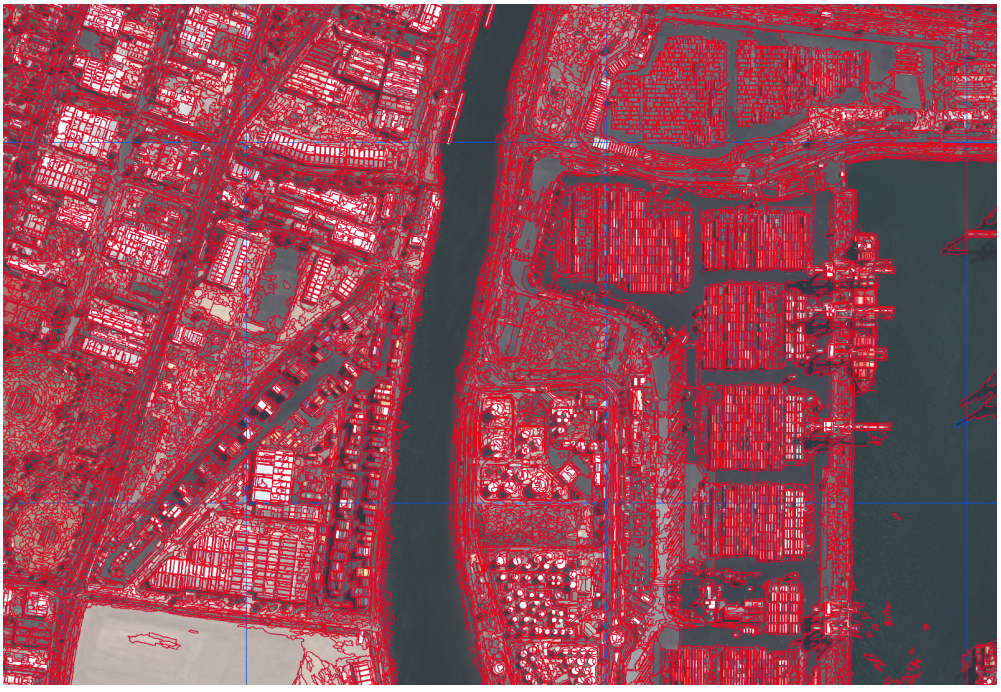
4. Item 4

   The following command will run the application in *vector* mode, with the *stitch* option enabled:

```
$ otbcli_Segmentation -in segmentation_large_xt_phr.tif
  -filter meanshift -filter.meanshift.ranger 30 -mode vector
  -mode.vector.out meanshift.sqlite -mode.vector.stitch 1
```

   Looking at the results in **QGis** one can see that most of the tiling effects have been removed by the stitching option (there might be some left). The results are therefore closer (but not identical) to what we would expect without the tiling strategy.

   Here is how the results look like in **QGis**. In blue, one can see the results without stitching, and in red, the results with stitching.



Homework

1. Item 1

   - The *tilesize* option allows to tune the size of the tile used during piecewise segmentation
   - The *simplify* option allows to simplify the output polygons up to a given tolerance (always expressed in pixels). The resulting file will be smaller.
   - The *minsize* option allows to discard segments whose size is smaller than a user-defined threshold (in pixels).

2. Item 2

   To avoid segmenting vegetation, one can build a vegetation mask with the **BandMath** application by thresholding the NDVI of the image. This mask can then be used in the segmentation application using the *mode.vector.inmask* option. Note that this mode is only available in *vector* mode.

3. Item 3

   Objects with high reflectance values are often more difficult to segment. Because of specular reflections, their inner variance is usually higher than other objects. Therefore, segmentation methods relying on comparison of neighboring pixels with respect to a given threshold will fail (this is the case for all three methods we used during the exercise).

   An idea to overcome this issue is to segment together all neighboring pixels with very high reflectance. This can be done with the connected components method, as shown earlier in the solution.

3.4 Feature extraction

3.4.1 Description

Abstract This workshop will introduce you to the **Feature Extraction** methods available in OTB. **Feature extraction**. Those methods are available in Monteverdi modules and also in applications integrated in Monteverdi2. You will learn how to use these modules and visually evaluate the usefulness of different features for image information extraction.

All exercises can be done using **Monteverdi** modules or **OTB applications**. Solutions for both software are provided.

Data If you need to generate the data used in this exercise from the original products, you can use the following command lines.

```
$ otbcli_ExtractROI \
  -in  FCGC600031035/IMG_PHR1A_MS_002/IMG_PHR1A_MS_201202250025599_SEN_PRG_FC_5847-002_R1C1.JP2 \
  -out phr_xs_melbourne.tif uint16 -startx 4096 -starty 2048 -sizex 4096 -sizey 4084
```

Pre-requisites

- Basic knowledge of **Monteverdi** usage
- Basic knwoledge of **OTB applications**

Achievements Being able to use the **Feature Extraction** modules of **Monteverdi** and **OTB applications**.

3.4.2 Steps

Getting around the Feature Extraction modules *Feature extraction* is a generic term which refers to the procedure of processing images in order to produce either new images or a set of objects in order to represent the information contained by the processed image in a higher level of abstraction.

Other terms than *features* can be used as for instance *primitives*, *indices* or *descriptors*. Although minor differences in meaning exist among these terms, we will use the as synonyms here.

In **Monteverdi** there are 5 different modules for feature extraction and they are all grouped under the *Filtering* → *Feature extraction* menu.

All *Feature extraction* modules follow the same rationale: they take an input image (with any number of channels) and produce an output image where each band is one of the computed features.

In this part of the exercise, we will use the following data:

`phr_xs_melbourne.tif`

1. Open the image in **Monteverdi**.
2. Open the **Smoothing** module from the *Filtering* → *Feature extraction* menu, and load the image and the segmentation inside the module.
3. What are the 3 image windows which appear at the top of the graphical user interface?
4. The *Action* tab is selected by default. What can you do on this part of the module?
5. Select the *Output* tab and describe what can you do on it.
6. What are the **OTB applications** which implement *feature extraction* algorithms.

Smoothing module You can either keep the *Smoothing* module open from the previous exercice or close it and open another one in order to start with a fresh one.

1. One of the available features is *Original data*. What is it and what may its purpose be?
2. Select the *Mean* feature and choose to compute it only on the *intensity* channel. What is the meaning of the *Radius along X* and *Radius along Y* parameters and what is their effect on the computed feature?
3. Select the *Meanshift smooth* feature. And try to guess the meaning of the parameters and their influence on the results.
4. Select the *Meanshift clusters* feature. As you can see, the same parameters as for the *Meanshift smooth* are available. Which is the difference in terms of the computed feature?

Edge extraction module Open the *Edge extraction* module and give it the input image. In order to keep thing simple, you will only work with one channel. Also, the *Harris detector* – which an interest point detector – and the *Touzi* filter – which is to be used with SAR images – will be ignored.

By reading text displayed on the *Feature Information* window and the list of available parameters and their effect on the computed features, answer to the following questions.

1. What does the *Meanshift boundaries* feature provide?
2. How can a *Variance* filter detect edges?
3. What is the effect of the *sigma* value on the *Recursive gradient* filter?
4. In the *Sobel* edge detection, the result is an edge density computed after the application of a Sobel edge enhancing filtering (high pas filter). You can set 2 different thresholds. What is their effect on the result?

Radiometric index extraction module By radiometric index we understand a combination of spectral bands which enhances a particular type of material. There are indices for vegetation, soils, water, artificial surfaces (built-up), and so on. You can find detailed descriptions and bibliographic references about this subjects by visiting the Radiometric indices page on the Orfeo Toolbox Wiki and following the links therein.

1. Open the *Radiometric index extraction* module and feed it with the 4-band Pléiades image. Select the *NDVI* index under the *Vegetation* list of indices. Generate the feature by using different selections on the *Channels Selection* list. Why is the result always the same regardless of the chosen channels?
2. Which are the spectral bands most commonly used for vegetation indices?
3. Find a value of the parameter s for the MSAVI index which gives results close to the ones obtained by the RVI. What is the interest of indices with parameters as this one with respect to indices like RVI or NDVI for which there are no parameters?
4. Which water indices can't be computed on a Pléiades image?

Texture extraction module

1. SFS The Structural Feature Set approach computes textures based on line direction analysis through the central pixel.

   Directions are computed at a constant step angle. A direction is defined as:

   $$d_i = \sqrt{(m^{e1} - m^{e2})^2 + (n^{e1} - n^{e2})^2}$$

   From d_i, histograms are defined:

   $$H(c) : \{c \in I \mid [d_1(c), \ldots, d_i(c), \ldots, d_D(c)]\}$$

   Thus, 6 textures are defined :

   $$length = \max_{i \in [1;D]} (d_i(c)$$

$$width = \min_{i \in [1;D]} (d_i(c)$$

$$PSI = \frac{1}{D} \sum_{1=1}^{D} d_i(c)$$

$$\omega - mean = \frac{1}{D} \sum_{1=1}^{D} \frac{\alpha.(k_i - 1)}{st_i} d_i(c)$$

$$ratio = \arctan \frac{\sum_{j=1}^{n} sort^j_{min}(H(c))}{\sum_{j=1}^{n} sort^j_{max}(H(c))}$$

$$SD = \frac{1}{D-1} \sqrt{\sum_{1=1}^{D} (d_i(c) - PSI)^2}$$

(a) Compute the *width* and the *length* features using the default parameters and explain what are the differences you observe on the results.

(b) Which is the effect of the spectral threshold on the computation of the features? Same question for the spatial threshold.

2. Haralick The Haralick textures are a set of indices computed from the grey-level co-occurrence matrices of an image. These are matrices computed on a grey-level image and for each pixel, a neighborhood is defined by a rectangular window which is shifted by an offset.

   Mathematically, a co-occurrence matrix C is defined over an nxm image I, parameterized by an offset $(\Delta x, \Delta y)$, as:

$$C_{\Delta x, \Delta y}(i,j) = \sum_{p=1}^{n} \sum_{q=1}^{m} \begin{array}{ll} 1, & \text{if } I(p,q) = i \text{ and } I(p+\Delta x, q+\Delta y) = j \\ 0, & \text{otherwise} \end{array}$$

   The image values are quantized using small number of bins so that the equality between pixels are likely to occur.

   There are 2 different implementations of the Haralick textures in Monteverdi. We will use the 1st one.

   (a) Given the description above, what do you think is the meaning of the *radius*, *offset*, *min/-max* and *quant. levels* parameters?

   (b) Compute the *Entropy* of the *intensity* channel for different radiuses (2, 3, etc.). Which is the effect of this parameter?

   (c) Compute the *Energy*, *Entropy*, *Correlation* and *Inertia* textures with the same parameters. Compare them and see if some of them are redundant.

3.4.3 Solutions

Getting around the Feature Extraction modules

1. Item 3 The graphical user interface displays a sub-sampled version of the input image onn the left, the full resolution on the center and the computed feature on the right.

2. Item 4 The *Action* tab allows you to select the feature to be computed, decide on which channels the feature will be computed (if applicable), set the parameters for the feature computation and see the list of computed features. The *Add* button effectively selects a given feature for computation.

3. Item 5 The *Output* tab allows you to decide which of the computed features will be kept in the output image and in which order they will be stored.

4. Item 6 **OTB applications** which implement *feature extraction* algorithms:

   - **LocalStatisticExtraction**
   - **Smoothing**
   - **MeanShiftSmoothing**
   - **EdgeExtraction**
   - **RadiometricIndices**
   - **BinaryMorphologicalOperation**
   - **GrayScaleMorphologicalOperation**
   - **HaralickTextureExtraction**
   - **SFSTextureExtraction**

   You can find more information about those applications in the Cookbook.

Smoothing module

1. Item 1 The *Original data* just copies the selected channels of the input image as bands of the output image. This can be useful in the case where you are creating a multi-channel image for a later classification and you want to include some of the original bands, side by side to other computed features.

2. Item 2 The radiuses define the size of a sliding window used to compute the mean around every pixel of the image. The window will be a rectangle centered on the pixel for which the mean is computed and sizes equal to $2 \times Radius_x + 1$ in the horizontal direction and $2 \times Radius_y + 1$ in the vertical one. The larger the radius the stronger the smoothing.

   You can also perform *Smoothing* with the following command:

   ```
   $ otbcli_Smoothing
     -in phr_xs_melbourne.tif
     -out phr_xs_melbourne_smooth.tif
     -type mean
     -type.mean.radius 2
   ```

3. Item 3 The *Meanshift smooth* uses the mean-shift algorithm to smooth the image. There are 2 main interests to this smoothing with respect to the classical mean seen on the previous point:

   (a) edge preservation;

   (b) can be used on multi-channel images and take profit of inter-channel correlation.

This algorithm performs the smoothing simultaneously on the image space (lines, columns) and on the feature space (for example, the 4-dimensional space defined by RGB+NIR images).

The meaning of the parameters is the following:

(a) Spatial radius: the radius of the spatial window used for the smoothing.

(b) Range radius: the radius of the smoothing window in the feature space.

(c) Min. region size: the minimum size for a region to be kept in the clustering step (not used for the smoothing).

(d) Scale: a multiplicative factor to be used for the image values which needs to be set if the image dynamics is low.

You can also perform *Smoothing* with the following command:

```
$ otbcli_MeanShiftSmoothing
  -in phr_xs_melbourne.tif
  -out phr_xs_melbourne_ms_smooth.tif
  -type mean
  -type.mean.radius 2
```

4. Item 4 The difference between the smoothing and the clustering is that the latter produces an image which is piecewise constant. That is, an image where connected pixels have the same value and form regions.

   These regions are defined at the end of the smoothing procedure by assigning each pixel the value of the mode of the histogram (in the feature space) to which it belongs. Since these histograms are computed also using a spatial window, the pixels belonging to the same mode are close pixels in space.

   When clusters (a set of connected pixels associated with the same histogram mode) define regions with sizes smaller than the minimum region parameter, they are merged with the closest and most similar one.

Edge extraction module

1. Item 1 It's just the boundaries of the regions produced by the *Meanshift clusters* feature of the *Smoothing* module.

2. Item 2 This filter assigns to each pixel the value of the local variance inside a window centered on it:

$$var(i,j) = \frac{1}{(2Radius_x+1)\times(2Radius_x+1)} \sum_{i-Radius_x}^{i+Radius_x} \sum_{j-Radius_y}^{j+Radius_y} (pix(i,j)-\mu(i,j))^2$$

   where $pix(i,j)$ is the input pixel value and $\mu(i,j)$ is the local mean computed using the same window.

   The variance values will be high when the pixel values inside the window deviate from the local mean. This can happen in 2 cases:

(a) When there is a strong texture effect.

(b) When there are 2 or more regions inside the window with different mean values. This is the case when an edge is present.

You can also compute local variance using the *LocalStatisticExtraction* with the following command:

```
$ LocalStatisticExtraction
  -in phr_xs_melbourne.tif
  -out phr_xs_melbourne_ms_variance.tif
  -channel 1
  -radius 2
```

3. Item 3 The recursive gradient uses a Gaussian smoothing (low pass filtering) previous to gradient computation for edge detection. The *sigma* parameter determines the width of the Gaussian smoothing, and therefore the degree of blurring applied to the image before gradient computation (edge detection).

   The effect of the *sigma* parameter will be the following: the larger the value, the wider the edges and the fewer the over-detections due to noise.

   Therefore, the choice of the value of *sigma* will depend on the noise level of the image and on the kind of edges that one wants to detect.

   You can compute the gradient magnitude (not the recursive gradient above) with the *EdgeExtraction*

```
$ EdgeExtraction
  -in phr_xs_melbourne.tif
  -out phr_xs_melbourne_ms_gradient.tif
  -filter gradient
  -radius 2
```

4. Item 4 The lower and upper thresholds define the intervals of pixels which will be set to 1 (below the lower and above the upper thresholds) or 0 (between the 2 thresholds) after the Sobel filtering and before the edge density computation. Therefore, the thresholds determine how the image produced by the Sobel filtering will be binarized before passing it to the density computation (percentage of detected pixels inside the window).

   You can compute the Sobel filter with the *EdgeExtraction* application:

```
$ EdgeExtraction
  -in phr_xs_melbourne.tif
  -out phr_xs_melbourne_ms_gradient.tif
  -filter sobel
  -radius 2
```


Radiometric index extraction module

1. Item 1 For the radiometric indices, the channel selection does not matter, since each index is a particular combination of spectral bands. The bands used are selected in the *Feature Parameters* group.

   You can generate NDVI using the **RadiometricIndices**:

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

2. Item 2 Most of the indices use the red (R) and the near infrared (NIR) bands, since the vegetation has a low response on the R and high on the NIR. Most indices use therefore combinations of differences and ratios of these bands.

   Sometimes, the green band is also used.

3. Item 3 Values greater than 6 should be fine.

   The interest of having parameters is being able to take into account soil reflectance for the cases of sparse vegetation. The *L* parameter of the SAVI index is close to 0 for very sparse vegetation and close to 1 for a very dense cover. The *s* parameter of the MSAVI index is the slope of the soil line, that is the NIR reflectance plotted as a function of the red reflectance for soil pixels.

   You can compute *MSAVI* index with the following command:

   ```
   $ otbcli_RadiometricIndices
     -in phr_xs_melbourne.tif
     -out phr_xs_melbourne_msavi.tif
     -channels.red 1
     -channels.green 2
     -channels.blue 3
     -channels.nir 4
     -list Vegetation:MSAVI
   ```

4. Item 4 The NDTI and the NDWI can't be computed on a Pléiades image (or a Quickbird image, for that matter) since the MIR (mid infrared, also callwed SWIR for short-wave infrared) is not available. In this case the NDWI2 index can be use.

   You can compute *NDWI2* index with the following command:

   ```
   $ otbcli_RadiometricIndices
     -in phr_xs_melbourne.tif
     -out phr_xs_melbourne_ndwi2.tif
   ```

```
-channels.red 1
-channels.green 2
-channels.blue 3
-channels.nir 4
-list Water:NDWI2
```

Texture extraction module

1. SFS

   (a) Item 1 It may seem contradictory, but the *width* feature gives hig values to pixels which belong to elongated regions, while the *length* feature gives brigth values to any region (elongated or not) which has a large area. If you have a look at the formulas for each feature you will understand why.

   You can compute *SFS* with the following command:

   ```
   $ SFSTextureExtraction
     -in phr_xs_melbourne.tif
     -out phr_xs_melbourne_ms_sfs.tif
     -channel 1
     -parameters.spethre 50.0
     -parameters.spathre 100
     -radius 2
   ```

   The available texture features are SFS'Length, SFS'Width, SFS'PSI, SFS'W-Mean, SFS'Ratio and SFS'SD.They are provided in this order in the output image.

   (b) Item 2 The spectral threshold sets the acceptable value of the difference between 2 adjacent pixels along a line in order to continue adding new pixels to the direction. Therefore, a small value for this thresholds will produce shorter lines and therefore fewer pixels with bright values.

   The spatial threshold stops the length of the line in the given direction regardless of the pixel values. Therefore, a low value for this threshold will also produce shorter lines.

2. Haralick

   (a) Item 1

   - The *radius* parameter determines the size if the local window used for the co-occurrence matrix computation.
   - The *offset* parameter sets the Δx and Δy values for the co-occurrence matrix.
   - The *min\/max* values can be used to define the range of image values over which the quantification levels will be defined.
   - The *quant. levels* parameter defines the number of discrete values that will be used for comparing the pixel values in the co-occurrence matrix.

   (b) Item 2 The larger the radius, the wider the detected areas, since we are introducing a kind of blurring of the computation by using larger windows.

   You can compute Haralick textures with the **HaralickTextureExtraction**:

```
$ HaralickTextureExtraction
  -in phr_xs_melbourne.tif
  -out phr_xs_melbourne_ms_haralick.tif
  -channel 1
  -parameters.xrad 2
  -parameters.yrad 2
  -parameters.xoffset 1
  -parameters.yoffset 1
  -parameters.min 100
  -parameters.max 4000
  -parameters.nbbin 8
```

(c) Item 3 Visually, *Energy* and *Entropy* seem to be the most correlated, since the pixel values are the most similar. However, if you have a closer look, you will see that all 4 textures give the same kind of information for typical remote sensign images. Although *Correlation* and *Energy* seem to be the most different because they present different contrasts, they enhance the same areas as the other textures.

Actually, Haralick textures are most useful for cases where pseudo-periodic patterns appear and the texture parameters are well suited. Otherwise, it is better to use 1st order statistics (as the local variance) which are much more easy to compute and yield the same kind of information.

3.5 Learning and classification from pixels

3.5.1 Description

Abstract This exercise will get you familiar with the OTB pixel based classification applications. You will learn how to train a SVM classification model from Pleiades images and a set of training regions. You will then learn how to apply this model to images and produce shiny classification maps.

Data If you need to generate the data `melbourne_ms_toa_ortho_extract_small.tif` used in this exercise from the original products (Bundle MS), you can use the following command lines.

To orthorectify the MS product:

```
 $ otbcli_OrthoRectification \
-io.in FCGC600031035/IMG_PHR1A_MS_002/IMG_PHR1A_MS_201202250025599_SEN_PRG_FC_5847-002_R1C1.JP2
-io.out ~/temporary/ortho_phr_ms_small.tif uint16 -outputs.mode auto -outputs.ulx 313892
-outputs.uly 5816639 -outputs.spacingx 2 -outputs.spacingy -2 -opt.ram 1024 -map utm
-map.utm.zone 55 -outputs.sizex  2048 -outputs.sizey 2048  -map.utm.northhem 0
```

To convert pixel values in top of atmosphere milli-reflectance:

```
$ otbcli_OpticalCalibration \
  -in ~/temporary/ortho_phr_ms_small.tif
  -out melbourne_ms_toa_ortho_extract_small.tif uint16 -milli 1 -level toa
```

You can also generate also the `melbourne_ms_toa_ortho_extract_large.tif` with the same set of commands. You just need to change the size of the orthorectify region to 4096 instead of 2048.

Pre-requisites

- Basic knowledge on OTB applications and QGIS usage
- Basic knowledge on image supervised classification
- Basic knowledge on GIS vector file formats

Achievements

- Usage of the OTB Classification applications
- Classification of large images
- Import of results in a GIS software

3.5.2 Steps

In this part of the exercise, you will use the following data:
`melbourne_ms_toa_ortho_extract_small.tif`

Produce and analyze learning samples

- Use QGIS to produce polygons for 5 classes (vegetation, roads, soil, buildings and water)
- Export this vector layer in ESRI Shapefile (.shp). Other OGR format can also be used here.
- What is the label corresponding to the class **water** in the shapefile? An example set of learning samples is provided for the exercise in *training.shp*

Tips and Recommendations:

- Note the field name of the shapefile which contains the label. You will need to provide this field in the training application

Estimate image statistics In order to make these features comparable between each images, the first step is to estimate the input images statistics. These statistics will be used to center and reduce the intensities (mean of 0 and standard deviation of 1) of training samples from the vector data produced by the user.

- Use the **ComputeImagesStatistics** to compute statistics on the image
- What is the mean of the red band?
- The extract provided has been converted from DN to milli-reflectance. For what reasons, is it advised to do so when performing multiple images classification?

Estimate classification model using the Support Vector Machine algorithm The **TrainImagesClassifier** application performs supervised classifier training from multiple pairs of input images and training vector data. Samples are composed of pixel values in each band optionally centered and reduced using XML statistics file produced by the **ComputeImagesStatistics** application. We will use this application with only one image in this exercise.

- How many classifiers are available in the **TrainImagesClassifier** application?
- Use the **TrainImagesClassifier** to produce SVM model
- Which kernel is used by default in the application?
- What is the measured accuracy?
- Use the **TrainImagesClassifier** to produce Random Forests model

Apply classification model

- Use the **ImageClassifier** to apply the classification model (SVM and RF) to the input image
- What is the output of the application?
- Bonus : Use the same model to apply the classification to the other extract

  `melbourne_ms_toa_ortho_extract_large.tif`

Produce printable classification map We are now going to produce a printable classification map using the **ColorMapping** application This tool will replace each label with an 8-bits RGB color specified in a mapping file. The mapping file should look like this :

```
$ # Lines beginning with a # are ignored
1 255 0 0
```

- Produce your custom look-up table (LUT)
- Use this LUT to produce a printable classification map for both classifier (in PNG format)
- Overlay this map on the input image in QGIS. Comment on the classification results.
-

Classification map regularization Resulting classification maps can be regularized in order to smoothen irregular classes. Such a regularization process improves classification results by making more homogeneous areas which are easier to handle.

- What is the application which allows to perform classification map regularization?
- What is the purpose of the boolean parameter *suvbool*?
- Perform classification map regularization using the output of the **ImageClassifier** application?

Fusion of classification maps After having processed several classifications of the same input image but from different models or methods (SVM, KNN, Random Forest,...), it is possible to make a fusion of these classification maps. The Fusion of Classifications generates a single more robust and precise classification map which combines the information extracted from the input list of labeled images.

- What application should be use to perform fusion of classification maps?
- How many algorithms are available in the application?
- Perform fusion of the 2 classification map using DS algorithm (fused SVM and RF classification maps)

Homework

- Produce classification model with different kind of SVM kernels and try other machine learning algorithms available. Comment different accuracies obtained?
- Going big: Apply this classification on the pan-sharpened image over Melbourne

3.5.3 Solutions

Produce and analyze learning samples The label corresponding to the **water** class in the shapefile is **2**

Estimate image statistics Here is the command line to produce the image statistics:

```
$ otbcli_ComputeImagesStatistics
  -il melbourne_ms_toa_ortho_extract_small.tif
  -out melbourne_ms_toa_ortho_extract_small_stats.xml
```

The value of the mean of the red band is **111.625**. Reflectance allows to compare radiometric values between images of different sensors.

Estimate classification model using the Support Vector Machine algorithm This application is based on LibSVM and on OpenCV Machine Learning classifiers, and is compatible with OpenCV 2.3.1 and later.

There is currently 9 algorithms available.

Here is the command line to produce the SVM model (default algorithm used by the application):

```
$ otbcli_TrainImagesClassifier
  -io.il melbourne_ms_toa_ortho_extract_small.tif
  -io.imstat melbourne_ms_toa_ortho_extract_small_stats.xml
  -io.vd training.shp
  -io.out melbourne_ms_toa_ortho_extract_small_model.svm
```

Linear kernel is used by default in the application. The measured accuracy is around 0.9. The value is high due to the low number of training samples and their lack of variability.

Apply classification model Here is the command line to produce the SVM model:

```
$ otbcli_ImageSVMClassifier
  -in melbourne_ms_toa_ortho_extract_small.tif
  -model melbourne_ms_toa_ortho_extract_small_model.svm
  -imstat melbourne_ms_toa_ortho_extract_small_stats.xml
  -out melbourne_extract_small_classification_5classes.tif
  uint8
```

Pixels of the output image will contain the class label decided by the SVM classifier.

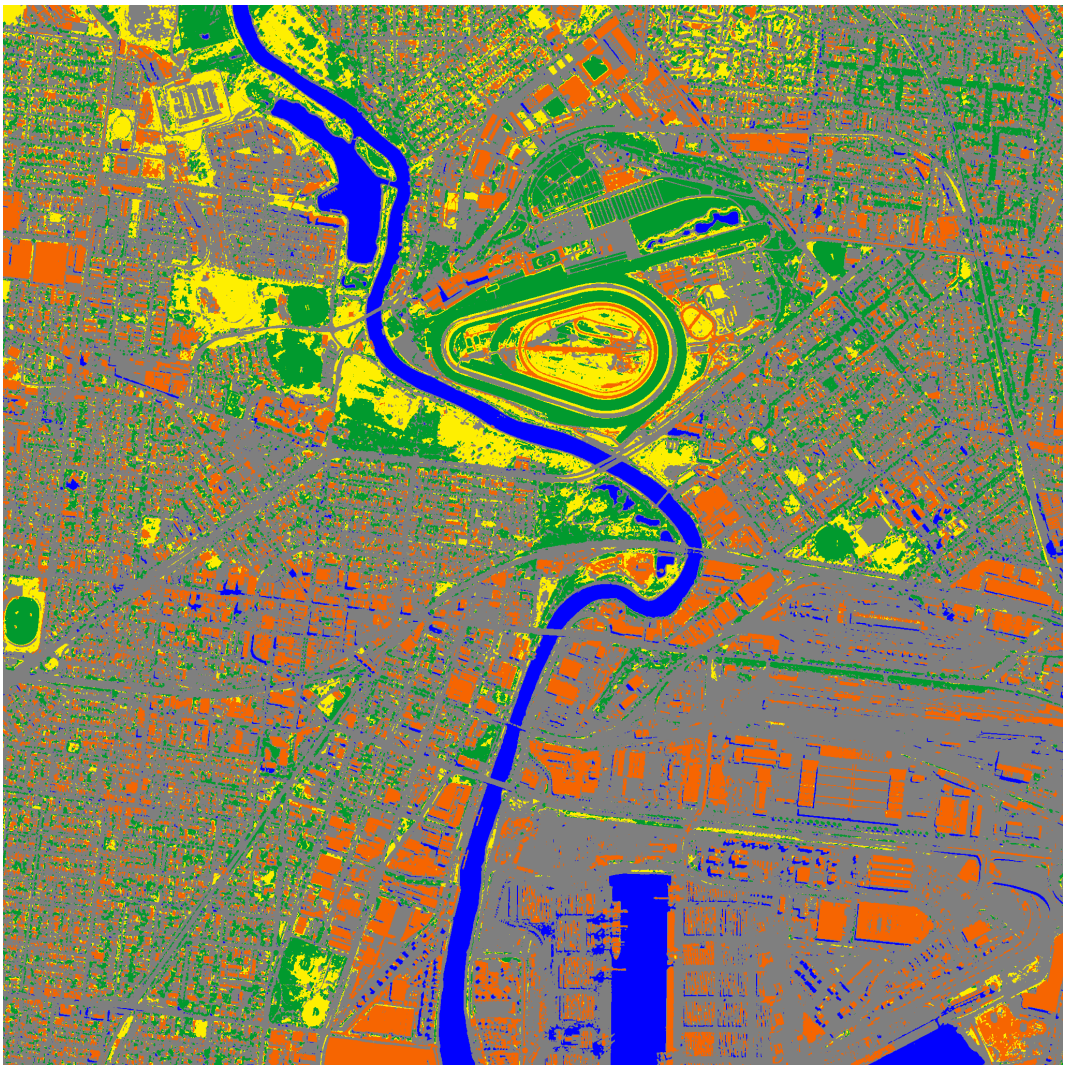
Produce printable classification map The look-up table (LUT) used to produce the classification map in my case is:

#Vegetation (green) 1 1 154 46 #Water (blue) 2 1 1 254 #Soil (yellow) 3 254 239 1 #Roads (grey) 4 127 127 127 #Buildings (orange) 5 246 101 1

Here is the command line to produce the classification map:

```
$ ColorMapping
  -in melbourne_extract_small_classification_5classes.tif
  -out melbourne_extract_small_classification_color.png uint8
  -method custom -method.custom.lut ColorTable.txt
```

And the result of the command:



Classification map regularization

1. Item 1 The application is the **ClassificationMapRegularization**.
2. Item 2 The **FusionOfClassifications** application uses either majority voting or the Demspter Shafer framework to handle this fusion.
3. Item 3

   Here is the command line to produce the fusion of classification maps:

```
$ FusionOfClassifications
  -il cmap1.tif cmap2.tif
  -out melbourne_extract_small_classification_fused.tif
  -method majorityvoting
  -nodatalabel    0
  -undecidedlabel 10
```

Fusion of classification

1. Item 1 The application is the **FusionOfClassifications**.
2. Item 2

   The boolean parameter *suvbool* used to choose whether pixels with more than one majority class are set to Undecided (true), or to their Original labels (false = default value). Please note that the Undecided value must be different from existing labels in the input image
3. Item 3

   Here is the command line to produce the classification map:

```
$ ClassificationMapRegularization
  -io.in melbourne_extract_small_classification_5classes.tif
  -io.out melbourne_extract_small_classification_regul.png uint8
  -ip.radius          3
  -ip.suvbool         true
  -ip.nodatalabel     10
  -ip.undecidedlabel  7
```

3.6 Learning and classification from objects

3.6.1 Description

Abstract This workshop will introduce you to the **Object Labeling** module of **Monteverdi**. You will learn how to use the module and see the influence of different features on classification results. You will also experiment with a simple active learning implementation on objects.

Data If you need to generate the data used in this exercise from the original products, you can use the following command lines.

```
$ otbcli_ExtractROI \
 -in ORTHO_UTM_PMS/IMG_PHR1A_PMS_001/IMG_PHR1A_PMS_201202250025599_ORT_IPU_20120504_1772-001_R1C1.JP2 \
 -out segmentation_small_xt_phr.tif uint16 -startx 11848 -starty 11426 -sizex 1024 -sizey 1024
```

Pre-requisites

- Basic knowledge of Object Based Image Analysis
- Basic knowledge on learning and classification

Achievements Being able to use the **Object Labeling** module of **Monteverdi**.

3.6.2 Steps

The preliminary segmentation In this part of the exercise, we will use the following data:

```
phr_pxs_melbourne_xt_small.tif
phr_pxs_melbourne_xt_small_segmentation.tif
```

1. Use the **ColorMapping** application to enhance the visualization of the segmented image (you can use the *optimal* and *image* modes as learned in the segmentation exercise).

2. Analyze the color-mapped segmentation results. For which kind of objects is the object based classification likely to work well ? For which kind of objects is it likely to perform badly?

Object Labeling module - basics

1. Open both the image and the segmentation image in **Monteverdi**.
2. Open the **Object Labeling** module from the *learning* menu, and load the image and the segmentation inside the module.
3. What is the purpose of each tab on the left side of the module?
4. In the *Objects* tab, create a new class. You can change its color and its name.
5. Right-click on an object of interest in the image. What happens?
6. Right-click a second time inside the selected object. What happens?
7. Add a few more objects to the current class.
8. Create a new class and add some objects to it.
9. Go to the *Features* tab, uncheck all features but the mean radiometric values.
10. Go to the *Learning* tab and click on classify. What happens?
11. Click on the *Save/Quit* button. What kind of outputs is produced by the module?

Tips and Recommendations:

- Choose two simple classes for this part of the exercise (for instance a *Water* class and a *Land* class)
- Use the navigation map to change the displayed area
- You can change the opacity of the classification layer as well as of the selected objects layer so as to better analyze the results.
- You can also clear the classification layer.

Object Labeling module - advanced In this part of the exercise, we will use these additional files: `samples.xml`.

1. Load again the image and the segmentation inside the module.
2. Load the samples file using *File/Load Samples*. What are the different object classes loaded ? How many samples per classes are used ?
3. Uncheck all features except from radiometric means:
   - Band1::Mean
   - Band2::Mean
   - Band3::Mean

- Band4::Mean

And use RBF SVM kernel with parameters optimization.myg

4. Perform the classification. What are the objects in the image that are badly classified because of missing classes ?
5. What are the objects in the image that are poorly classified because they are badly segmented or too complex ?
6. Try to enhance the classification by adding missing classes.
7. Try to enhance the classification by adding new features.

Tips and Recommendations:

- The **Object Labeling** module is quite memory consuming. Depending on the available memory on your system, you might want to restart **Monteverdi**.

Object Labeling module - active learning

1. In the *Objects* tab, click on the *Sample* button in the lower-left area. This will show you difficult samples by using the *margin sampling* technique.
2. What kind of segments are considered by the algorithm as hard to classify ?
3. Try to create a *Trash* class to handle noise segments.
4. Perform a few more iteration of active learning. What do you observe ?

3.6.3 Solutions

The preliminary segmentation

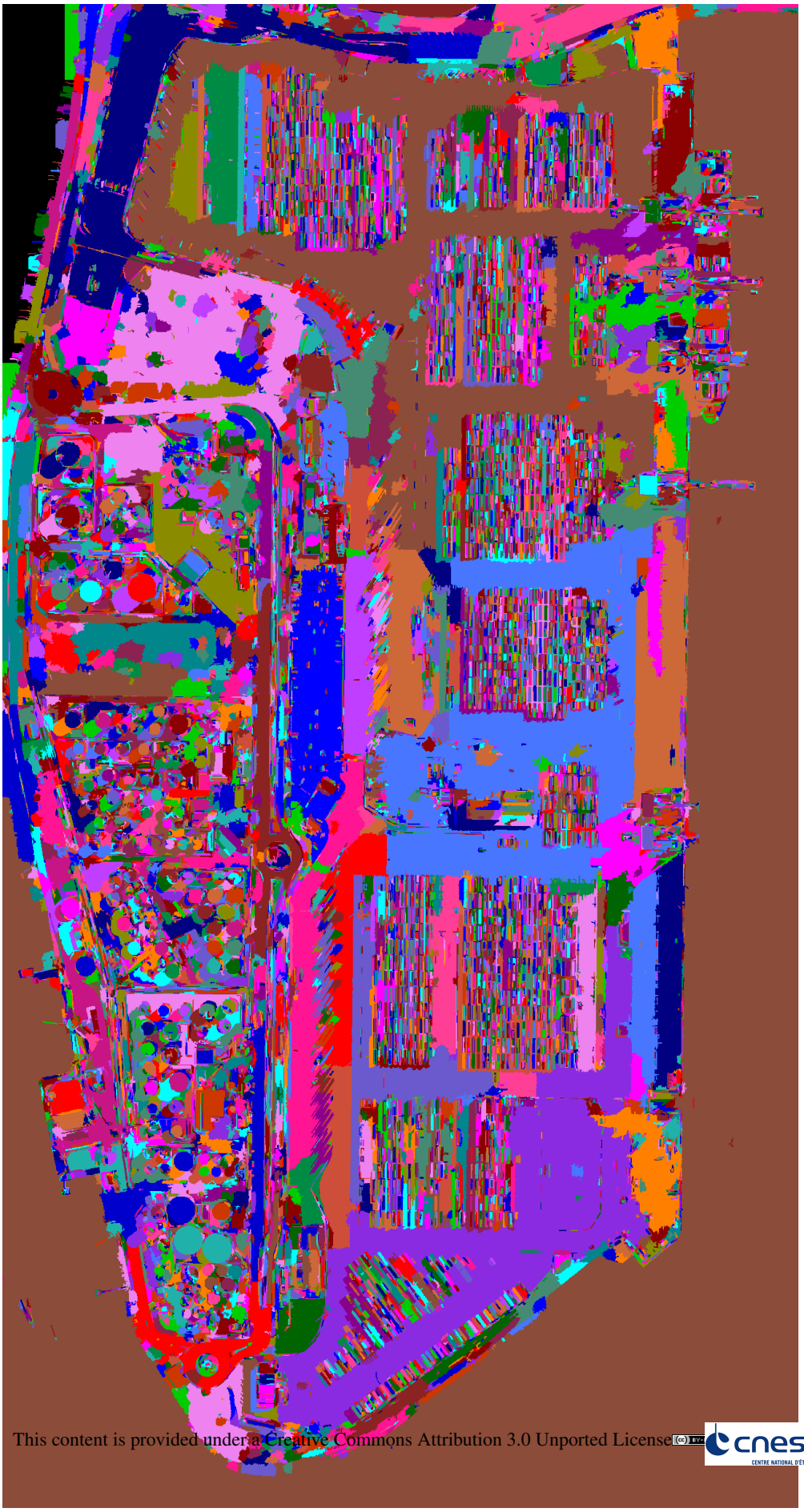
1. Item 1

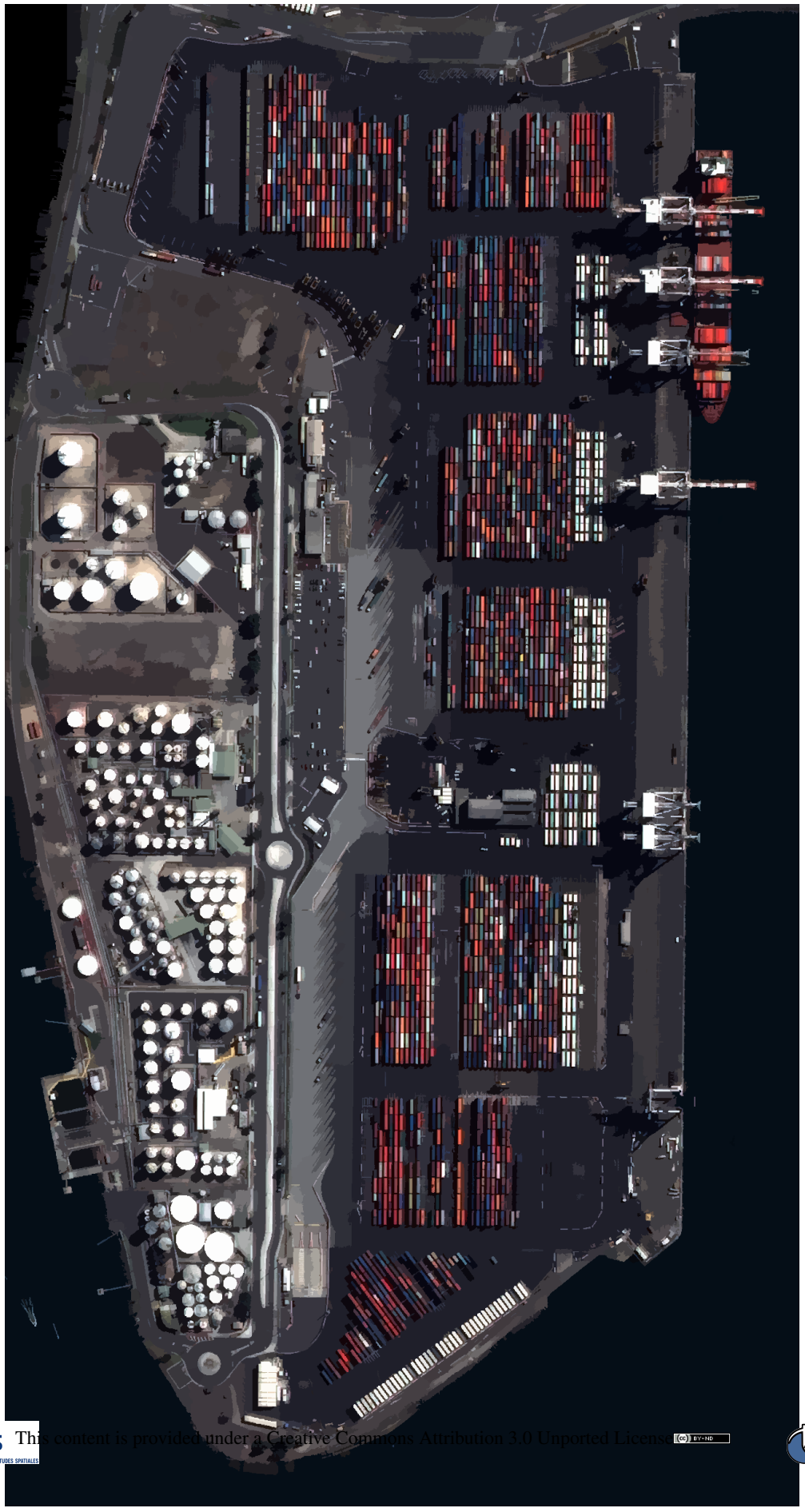
   Here are the command-line to generate the color-mapped images:

```
$ otbcli_ColorMapping
  -in phr_pxs_melbourne_xt_small_segmentation.tif
  -out obc_segmentation_optimal.png uint8
  -method optimal
```

```
$ otbcli_ColorMapping
  -in phr_pxs_melbourne_xt_small_segmentation.tif
  -out obc_segmentation_image.png uint8
  -method image
  -method.image.in  phr_pxs_melbourne_xt_small.tif
  -method.image.low 0
  -method.image.up 0
```

Here is what the color-mapped images look like:





2. Item 2

   From the segmentation results, we can infer that an object-based classification method might perform well on:

   - Most of circular containers,
   - Most of rectangular containers,
   - Simple classes like water or roads.

   But it will most likely fail on:

   - Some circular or rectangular containers that are fragmented by segmentation,
   - Complex objects like the boat or the cranes
   - Small objects like cars and trucks.

Object Labeling module - basics

1. Item 3

   The *Objects* tab allows to create classes and to add training segments to these classes. The *Features* tab allows to select the object-based features to be used for classification. Last, the *Learning* tab allows to tune classification parameters and to perform the classification.

2. Item 5

   The segment (from the image segmentation) under the mouse pointer gets selected on first right-click action.

3. Item 6

   The selected segment is added to the current class on second right-click action.

4. Item 10

   A SVM classifier is trained according to created classes and corresponding training samples, and the remaining of the image segments are classified using the trained classifier.

5. Item 11

   When the *Save/Quit* button is pressed, the module closes and produces three different outputs:

   - An image of labels corresponding to the classes,
   - A color-mapped image according to classes colors,
   - A vector outputs containing polygons labeled with their predicted classes.

Object Labeling module - advanced

1. Item 2

   The classes selected in the samples file are:

   - Circular containers
   - Rectangular colored containers

- Rectangular white containers
- Water
- Asphalt

2. Item 4 Using the provided samples and parameters, we get the following result. We can see that some basic classes are detected at the expense of more misclassification on difficult objects, as shown in left part of the figure at the end of this section.

   We can see some obvious missing classes in the training set leading to classification errors:

   - Shadows area get classified as Water. Even if Shadow is not a strictly-speaking a class of interest, the overall classification quality would benefit from a Shadow class.
   - Vegetation areas, even if there are only few of them in the images, also get miss-classified because there is no vegetation class in the training set.

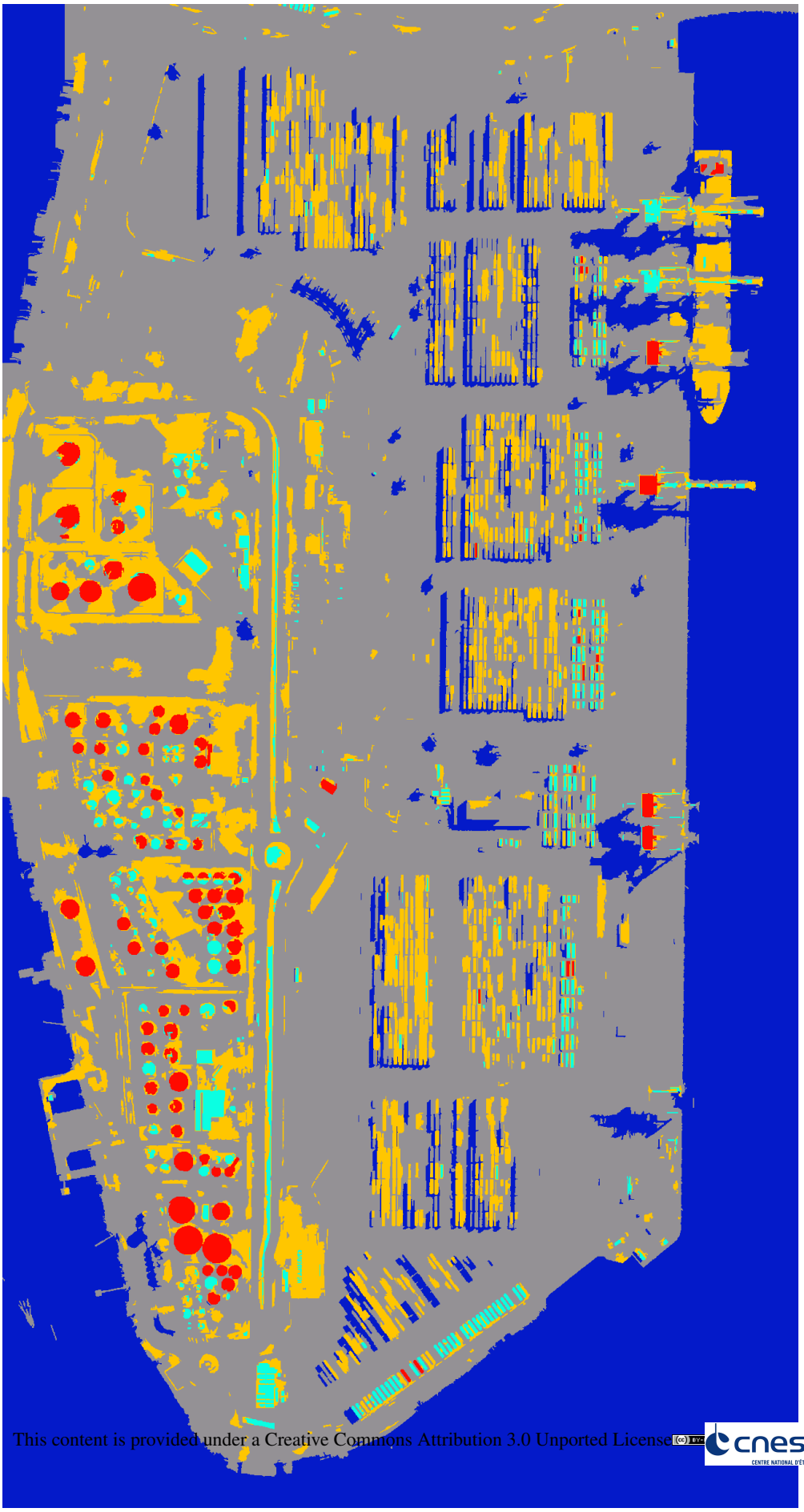
3. Item 5

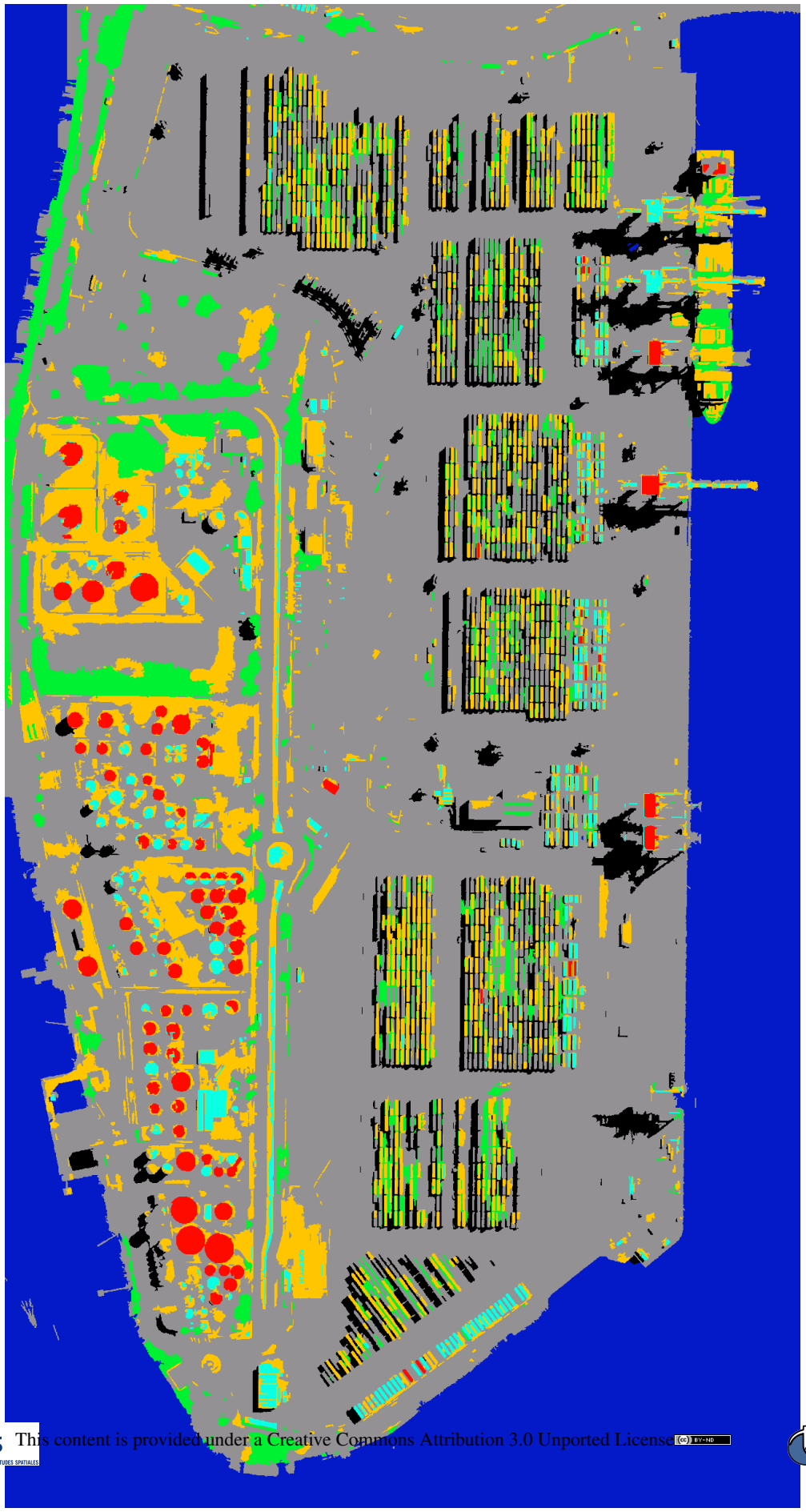
   As foreseen in section 3.6.3, some objects of interest are poorly segmented or too complex for good classification results:

   - The boat and the cranes are too complex,
   - Some of the containers (spherical or rectangular) are poorly segmented, which leads to miss-classification.

4. Item 6 - 7

   By adding a few more classes and samples, we get the result presented on the right of following figure.





Object Labeling module - active learning

1. Item 2

   The implemented active learning strategy often shows objects that are difficult to label manually, because they correspond to parts of fragmented objects or to segmentation noise.

2. Item 4

   We can observe that occasionally, the active learning strategy will discover a new kind of object, for which no class has been created yet. It may also run several times into the same objects that are still difficult to classify after some iterations.

3.7 Elevation map from stereo pair

3.7.1 Description

Abstract This exercise will guide get you familiar with the set of OTB applications which allow to compute elevation map from a stereo pair of optical images. You will learn how to :

- re-sample for stereo pair in epipolar geometry to reduce the stereo correspondences to a 1D problem
- Perform block matching between the 2 images to extract the disparity (related to the elevation)
- Filter disparities using correlation metric analysis

Data If you need to generate the data used in this exercise from the original products, you can use the following command lines.

```
$ otbcli_ExtractROI \
  -in PRIMARY_TRISTEREO_BUNDLE/IMG_PHR1A_P_001/IMG_PHR1A_P_201202250026276_SEN_IPU_20120509_2001-006_R1C1.JP2 \
  -out  tristereo_melbourne_1_small.tif uint16 -startx 25036 -starty 12455 -sizex 1024 -sizey 1024

$ otbcli_ExtractROI \
  -in  PRIMARY_TRISTEREO_BUNDLE/IMG_PHR1A_P_003/IMG_PHR1A_P_201202250025329_SEN_IPU_20120509_2001-008_R1C1.JP2 \
  -out tristereo_melbourne_2_small.tif uint16 -startx 25020 -starty 11863 -sizex 1024 -sizey 1024
```

Pre-requisites

- Basic knowledge on OTB applications
- Basic knowledge on epipolar geometry. Epipolar geometry is the geometry of stereo vision (see here). The operation of stereo rectification determines transformations to apply to each image such that pairs of conjugate epipolar lines become collinear, parallel to one of the image axes and aligned. In this geometry, the objects on a given row of the left image are also located on the same line in the right image.

- Basic knowledge of stereoscopic reconstruction

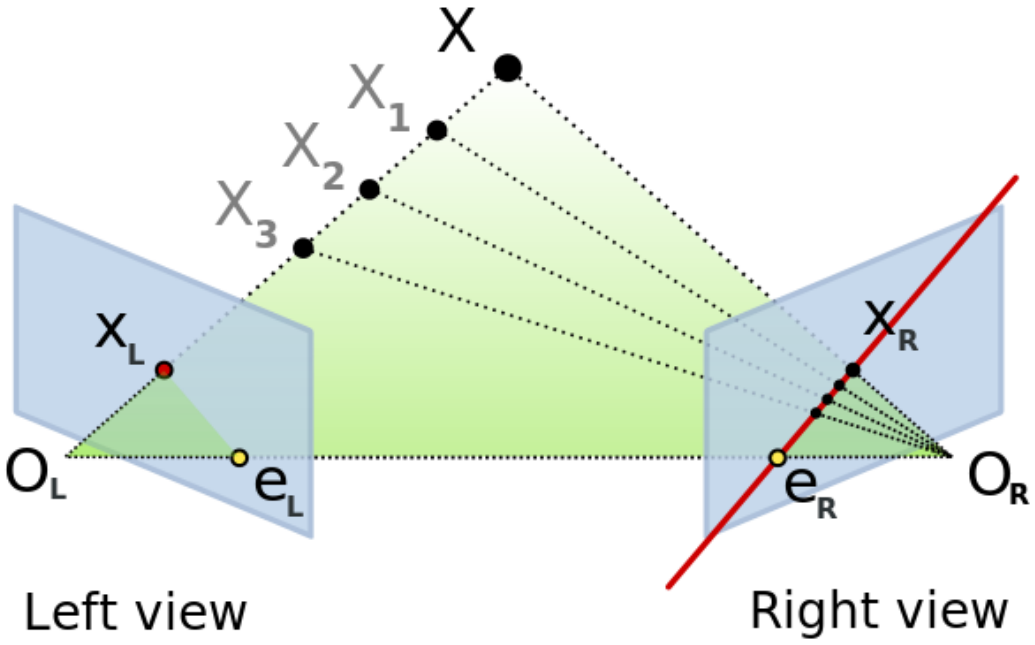


Figure 3: Epipolar geometry

Achievements

- Usage of stereoscopic reconstruction applications
- Stereo reconstruction based on Pleiades stereo images pair

3.7.2 Steps

From images to epipolar geometry In this part of the exercise, you will use the following data: `tristereo_melbourne_1_small.tif and tristereo_melbourne_2_small.tif`

1. Run the command-line and graphical version of the **StereoRectificationGridGenerator** application
2. What are the two outputs of the applications?
3. Use the application to generate two re-sampling grids. Which OTB application allows to re-sample the two input images using these grids?
4. Use this application to resample input stereo pair in epipolar geometry, open the 2 re-sampled images. What do you see ?

Tips and Recommandations:

- Perform the grids estimation using and average elevation of 20.45m (**epi.elevation.avg.value** keyword)

- Stereo-rectification deformation grid only varies slowly. Therefore, it is recommended to use a coarser grid (higher step value) in case of large images (**epi.step** keyword)
- Note the size of the images in epipolar geometry (output by the application)

Improvement of epipolar geometry Pleiades data can be orthorectified to absolute accuracies of about 10 meters, as a consequence there is still a need to improve the geometric accuracy (this is the case for all satellite imagery). For orthorectification purpose, it is achieved by optimizing sensor modelling with Ground Control Points. An other way to do this in case of superimposition of multiple images is to produce homologous points on each images and refine with these points the co-localisation function. It allows to improve the geometric accuracy and to produce consistent epipolar images.

We provide for the next questions a refined version of the stereo pair:
`tristereo_melbourne_1_small_ref.tif`
`tristereo_melbourne_2_small_ref.tif`

1. Recompute epipolar geometry with the new stereo pair (post-fixed by `_ref.tif`). Open the 2 versions of epipolar couples (total of 4 images). What differences do you notice between the two images pair?
2. Combined the 2 images to create a 3D anaglyph (left image on the red channel and the right image on the green and blue channel). Visualize the anaglyph with anaglyph glasses.

We will use this images in the next questions.

Block matching We are going to perform stereo pair block matching on the two images using the **BlockMatching** application.

1. Run the command-line and graphical version of the **BlockMatching** application. What are the mandatory parameters?
2. Propose manual or automatic methodologies to estimate the interval of disparities in vertical or horizontal direction.
3. Use these parameters to generate a disparity map and open the result with Monteverdi. What do you notice?

   Tips and Recommendations:

   - Discard pixels with no-data (0 in our case) value using the parameter **-mask.nodata**

Advanced Block matching : refine disparity map We are going to try now to improve the quality of the disparity map using options available in the **BlockMatching**.

1. Use the Normalized Cross Correlation and output the metric value using the io.outmetric option. Open the metric image, which values of correlation corresponds to a good disparity value ?
2. Use the option mask.variancet to discard pixels whose local variance is too small (the size of the neighborhood is given by the radius parameter)
3. Use the **BandMath** application to only keep horizontal disparity with high correlation value.

From disparity map to ground elevation Use the **DisparityMapToElevationMap** to transform the disparity map into an elevation map.

1. At which height approximately do cricket players play in the stadium?
2. What is approximately the height of the stadium?

Tips and Recommandations:

- Reuse the same average elevation of 20.45m
- Bonus : produce a mask using the **BandMath** application to discard pixels with low correlation values using the parameter **io.mask**

Homework

1. Try refinement steps to improve epipolar geometries (available soon in OTB -> 3.16 version)
2. Perform disparity coherence analysis by comparing disparity maps obtained by switching the left and right images
3. Re-compute disparity maps with sub-pixel precision block-matching
4. Use median filter to get a smoother disparity map

3.7.3 Solutions

From images to epipolar geometry

1. Item 1 To get the command-line help, run

   ```
   $ otbcli_StereoRectificationGridGenerator
   ```

   To get the graphical version of the **StereoRectificationGridGenerator** application, run

   ```
   $ otbgui_StereoRectificationGridGenerator
   ```

2. Item 2 The application estimates the displacements to apply to each pixel in both input images to obtain epipolar geometry.
3. Item 3 The **GridBasedImageResampling** application allows to resample the two input images in the epipolar geometry using these grids. These grids are intermediary results, not really useful on their own in most cases.

   ```
   $ otbcli_StereoRectificationGridGenerator
     -io.inleft tristereo_melbourne_3_small_ref.tif
     -io.inright tristereo_melbourne_1_small_ref.tif
     -io.outleft 31_grid_tristereo_melbourne_3_small_ref.tif
     -io.outright 31_grid_tristereo_melbourne_1_small_ref.tif
     -epi.elevation avg -epi.elevation.avg.value 20.45
   ```

4. Item 4

   For the left image :

```
$ otbcli_GridBasedImageResampling
  -io.in tristereo_melbourne_3_small_ref.tif
  -io.out 31_epi_tristereo_melbourne_3_small_ref.tif
  -grid.in 31_grid_tristereo_melbourne_3_small_ref.tif
  -out.sizex 1237 -out.sizey 1237
```

   For the right image:

```
$ otbcli_GridBasedImageResampling
  -io.in tristereo_melbourne_1_small_ref.tif
  -io.out 31_epi_tristereo_melbourne_1_small_ref.tif
  -grid.in 31_grid_tristereo_melbourne_1_small_ref.tif
  -out.sizex 1237 -out.sizey 1237
```

Refinement of epipolar geometry

1. Item 1

   The epipolar couple generated with the images with refined geometry does not present disparities in the vertical direction.

2. Item 2

   Here is the command-line to run the **ConcatenateImages** application to generate the anaglyph image:

```
$ otbcli_ConcatenateImages -il
  31_epi_tristereo_melbourne_3_small_ref.tif
  31_epi_tristereo_melbourne_1_small_ref.tif
  31_epi_tristereo_melbourne_1_small_ref.tif
  -out 31_anaglyph_3_1.tif
```

   Here is the result of this command:

Block matching

1. Item 1 The mandatory parameters are the intervals of disparity in the horizontal and vertical direction. In our case the interval in vertical direction should be void.

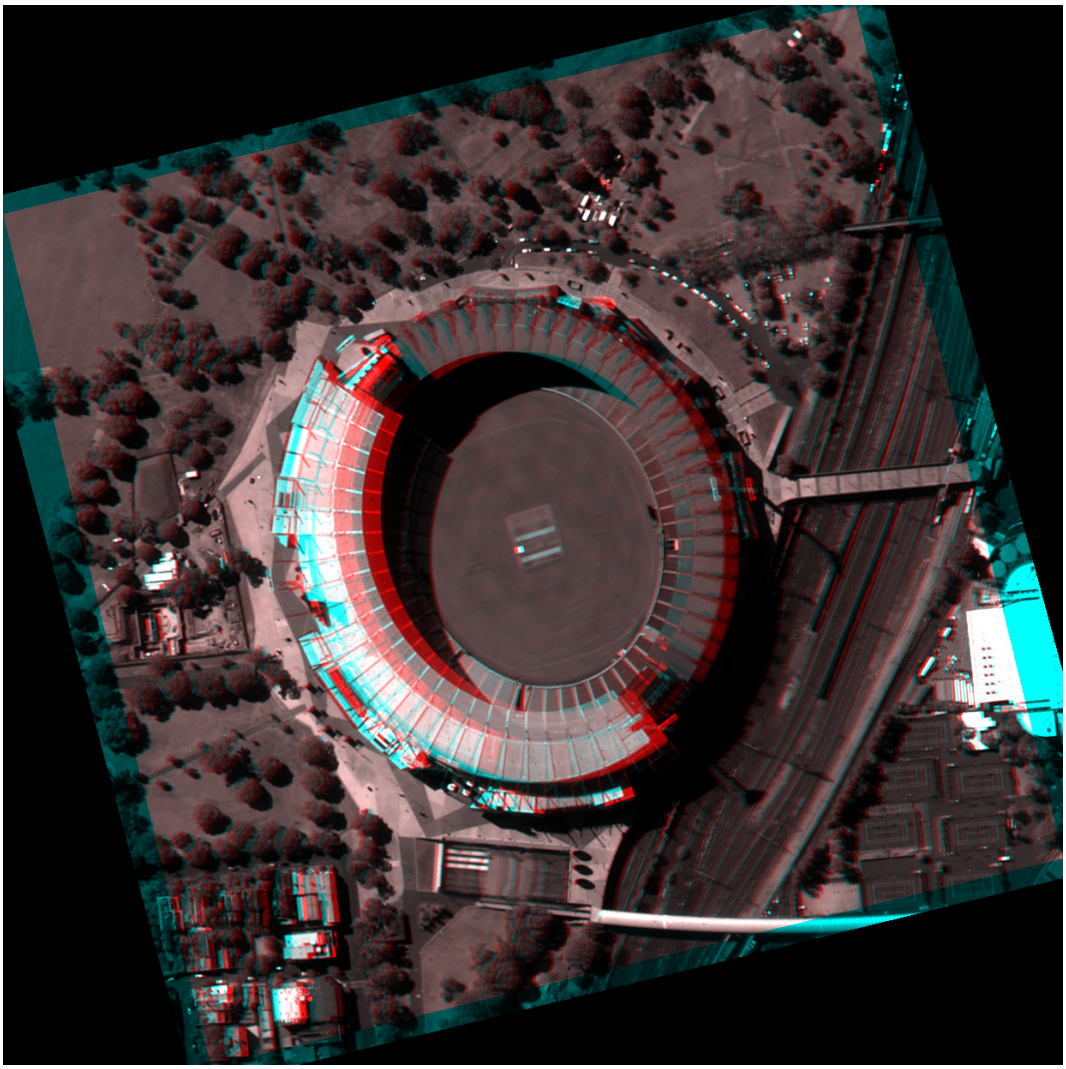


Figure 4: Epipolar geometry

2. Item 2 In theory, the block matching can perform a blind exploration and search for a infinite range of disparities between the images of the stereo pair. We need now to evaluate a range of disparities where the block matching will be performed.

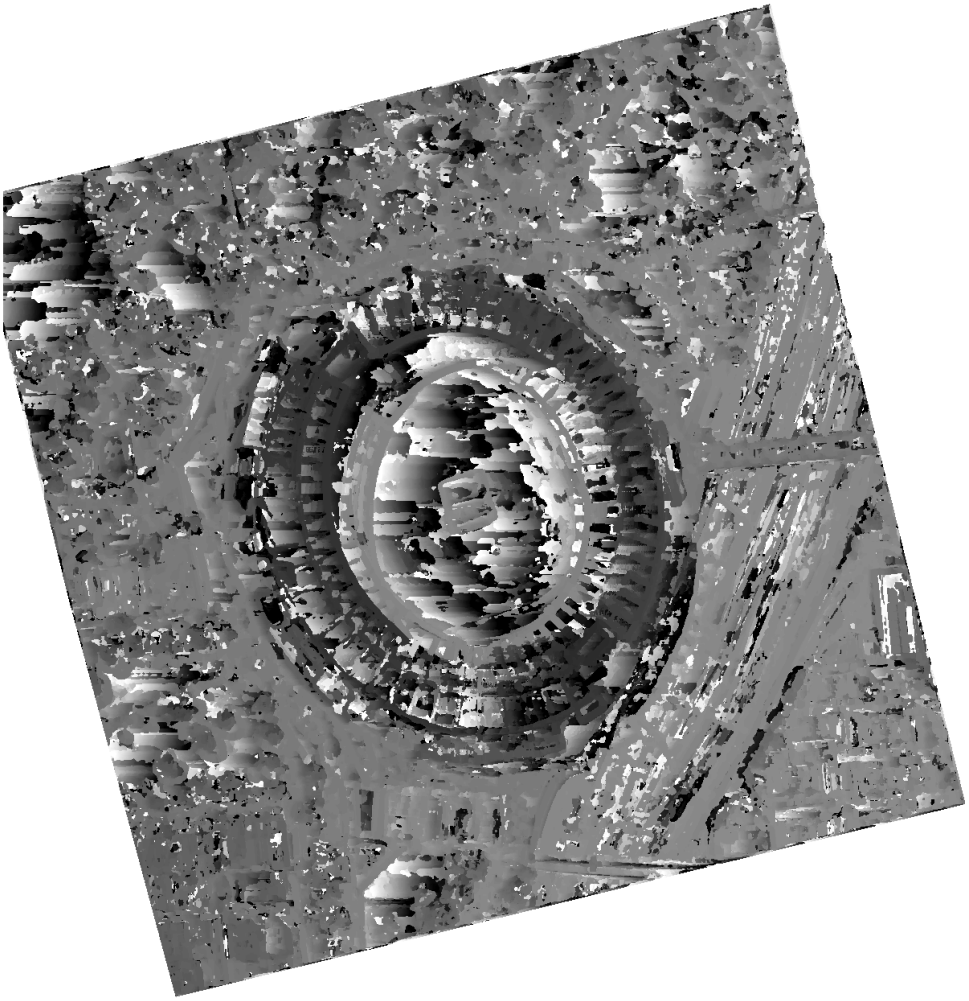
   In our case, we take one point on a *ground* area. The image coordinate in the first image is is $[275, 343]$ and in the second image is $[277, 343]$. We then select a second point on a higher region (in our case a point near the top of the Melbourne Cricket Ground) The image coordinate of this pixel in the first image is $[712, 354]$ and in the second image is $[671, 354]$. We can see that for the horizontal exploration, we must set the minimum value lower than -41 and the maximum value higher than 2 (pay attention to the convention for the sign of the disparity, the range is defined from the left to the right image).

3. Item 3

   Here is the command-line to run the application with default parameters:

```
$ otbcli_BlockMatching
  -io.inleft 31_epi_tristereo_melbourne_3_small_ref.tif
  -io.inright 31_epi_tristereo_melbourne_1_small_ref.tif
  -io.out 31_disparity_map_3_1.tif
  -bm.minhd -40 -bm.maxhd 40 -bm.minvd 0 -bm.maxvd 0
```

and here the result of this command:



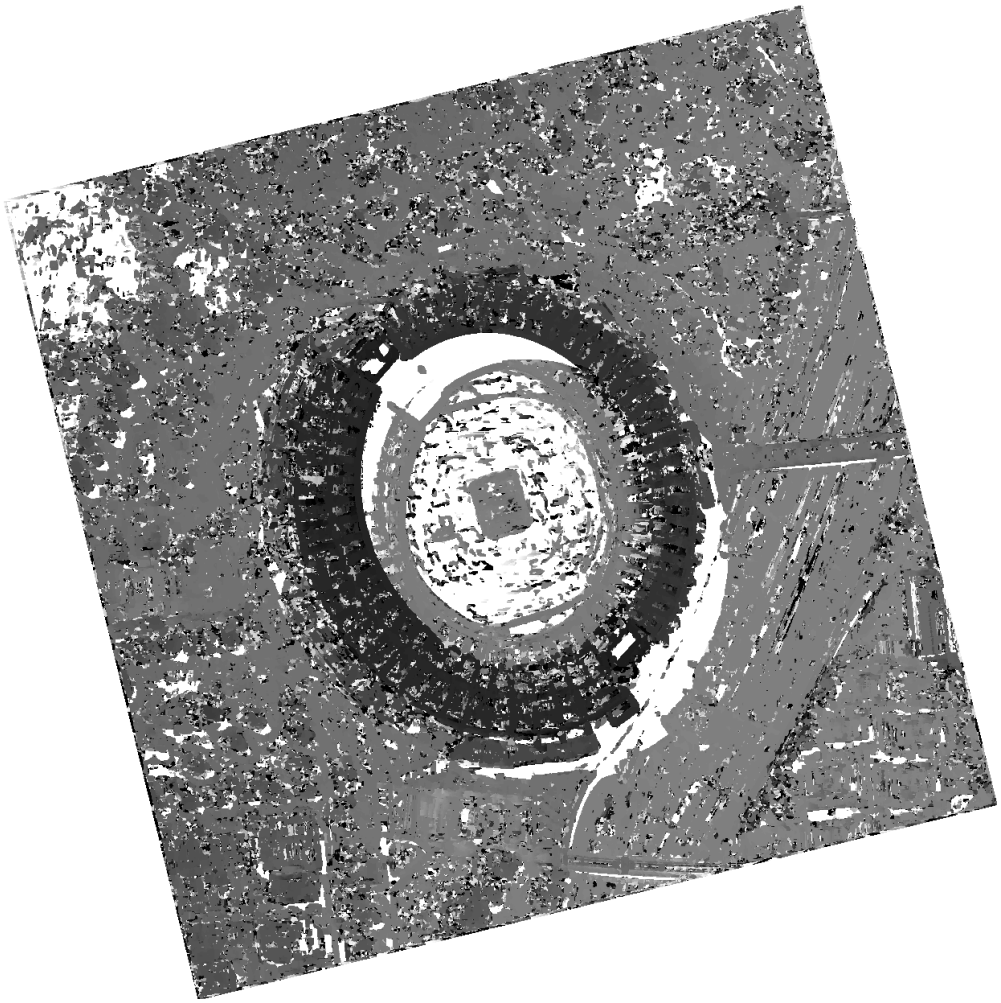
It shows that we need to discard pixels where block matching does not work and also filter low correlation values.

Advanced Block matching: refinement of the disparity map

1. Item1 Use the following parameters: **-io.outmetric 1 -bm.metric ncc**
2. Item2 Use the **mask.variancet** parameter.
   Here is the command-line to run the application witch combine all these parameters:

```
$ otbcli_BlockMatching
  -io.inleft 31_epi_tristereo_melbourne_3_small_ref.tif
  -io.inright 31_epi_tristereo_melbourne_1_small_ref.tif
  -io.out 31_disparity_map_3_1.tif
  -bm.minhd -40 -bm.maxhd 40 -bm.minvd 0 -bm.maxvd 0
  -mask.nodata 0 -mask.variancet 100 -io.outmetric 1
  -bm.metric ncc
```

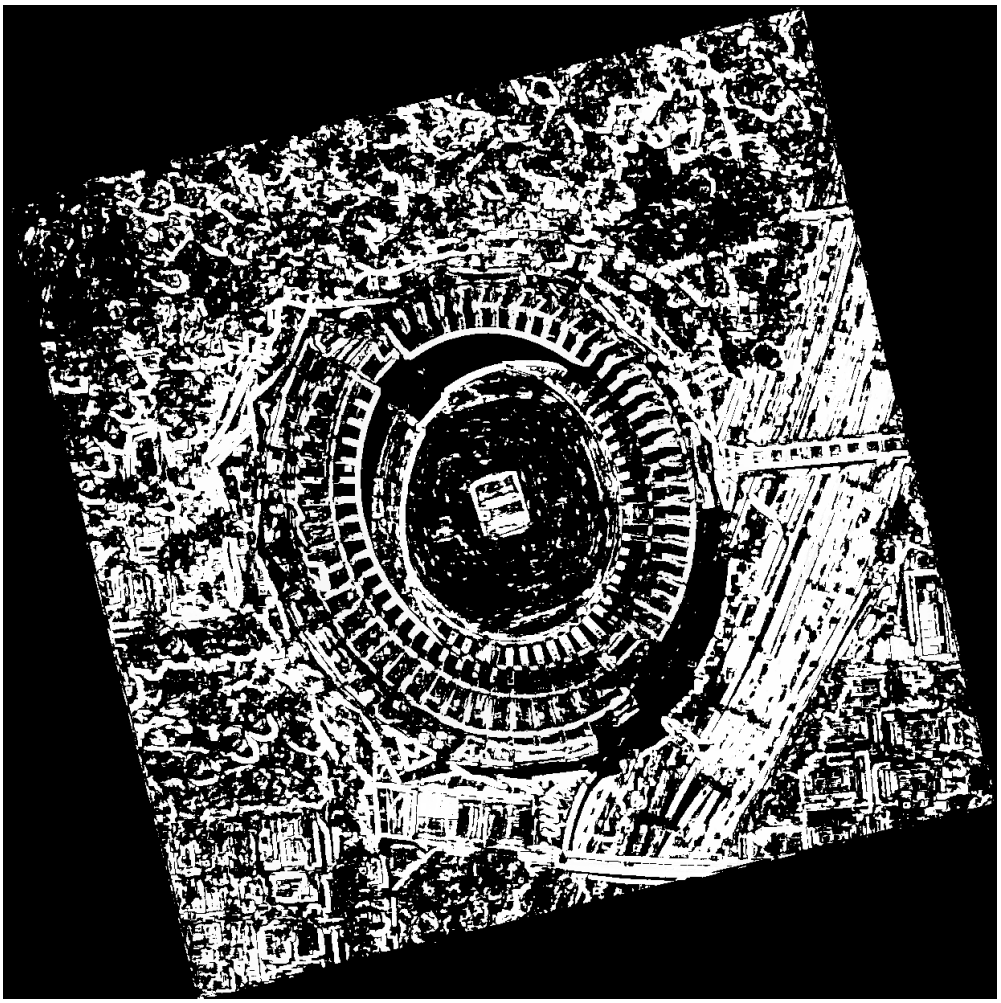
Here is the result of this command:



3. Item3

```
$ otbcli_BandMath
  -il 31_disparity_map_3_1.tif
  -out 31_filtered_disparity_map_3_1.tif
  -exp "if(im1b3>0.9,im1b1,-1000)"
```

Here is the result of this command:



From disparity map to ground elevation

1. Item1

   Here is the command-line to run the application:

```
$ otbcli_DisparityMapToElevationMap
  -io.in 31_disparity_map_3_1.tif
  -io.left tristereo_melbourne_3_small_ref.tif
  -io.right tristereo_melbourne_1_small_ref.tif
  -io.lgrid 31_grid_tristereo_melbourne_3_small_ref.tif
  -io.rgrid 31_grid_tristereo_melbourne_1_small_ref.tif
  -hmin 0 -hmax 80 -elev average -step 1
  -elev.average.value 20.45
  -io.out 31_disparity_map_to_elevation_3_1.tif
```

Here is the result of the command:

2. Item2 I found 20 meters for the ground and 58m for the roof. See this Wikipedia article for ground truth.

3.8 SAR Image Processing

3.8.1 Description

Abstract This workshop will introduce you to SAR image processing by using **Monteverdi**. You will learn how to visualize complex SAR images, how to reduce noise and how to detect edges.

Pre-requisites Basic knowledge of **Monteverdi** usage.

Achievements Being able to manipulate SAR images using **Monteverdi**.

3.8.2 Steps

Introduction to SAR images In this part of the exercise, we will use an extract of TerraSAR-X image called `2008-03-10_GrandCanyon_SSC` for which the principal file is:

`TSX1_SAR__SSC_____SM_S_SRA_20080310T133220_20080310T133228.xml`

The extract corresponds to a region of interest of size 1024×1024 pixels starting at coordinates (upper left corner) $(1800, 28000)$.

In this part of the exercise, we will use the following data:

`extract_tsx_canyon_complex.tif`

You can generate this extract using the `gdal_translate` utility with the following command:

```
$ gdal_translate
  TSX1_SAR__SSC_____SM_S_SRA_20080310T133220_20080310T133228.xml
  extract_tsx_canyon_complex.tif
  -in phr_xs_melbourne.tif
  -srcwin 1800 28000 1024 1024
```

1. Open the image in **Monteverdi** by choosing the above-mentioned XML file. Describe what you get in the available image list on the **Monteverdi** GUI.

2. Use the viewer to display the image. Add the 4 *parts* of the image to the same viewer so that you can switch between them in the same viewer display. Move the full resolution view to the center of the image. Compare the contents of the 4 available bands and answer to the following questions:

   (a) Which is the image which gives the point of view the most interesting about the observed landscape?

   (b) Describe the shapes of the histograms of the images.

3. Use the *BandMath* module to compute the image intensity.

4. In the following exercices, we will only use the intensity image. You can prepare a small image by extracting a region of interest of size 1024×1024 pixels starting at coordinates (upper left corner) $(1800, 28000)$.

Speckle reduction SAR images are strongly affected by speckle, a particular kind of noise present in all coherent acquisition systems (sonar, laser, etc.). This noise is very strong and it has a multiplicative behaviour.

Monteverdi offers 2 speckle reduction filters which are available under the *SAR/$→$\$/Despeckle* menu. In this exercice, we will use the Frost filter, which has 2 parameters: the radius – window size – and the *deramp* parameter which regulates the decay of an exponential function which is used as a weight for the pixels inside the window. The further the pixels are from the center pixel, the lesser they contribute to the computation of the filtered value for the center pixel.

1. Effect of the window size: perform the Frost despeckle with 3 different radiuses (3, 5 and 10) and comment on the differences between the filtered images. As a reference for comparisons, use the square shaped area located around the coordinates $(250, 300)$ of the extracted region of interest.

2. Comment on the shape of the histograms of the filtered images as a function of the radiuses. Compare these histograms to the one of the original image (before filtering).

3. Effect of the *deramp* parameter: keep a fixed radius of 10 and generate 3 filtered images by using the following values for the deramp parameter: 0.05, 0.2 and 0.8. Comment on the effects on the results.

Edge detection on SAR images warning : need a vector image so ROI is concatenated with itself! bug in Touzi filter where radius is not taken into account!

Image clustering compute textures and cluster idem with filterd images

3.8.3 Solutions

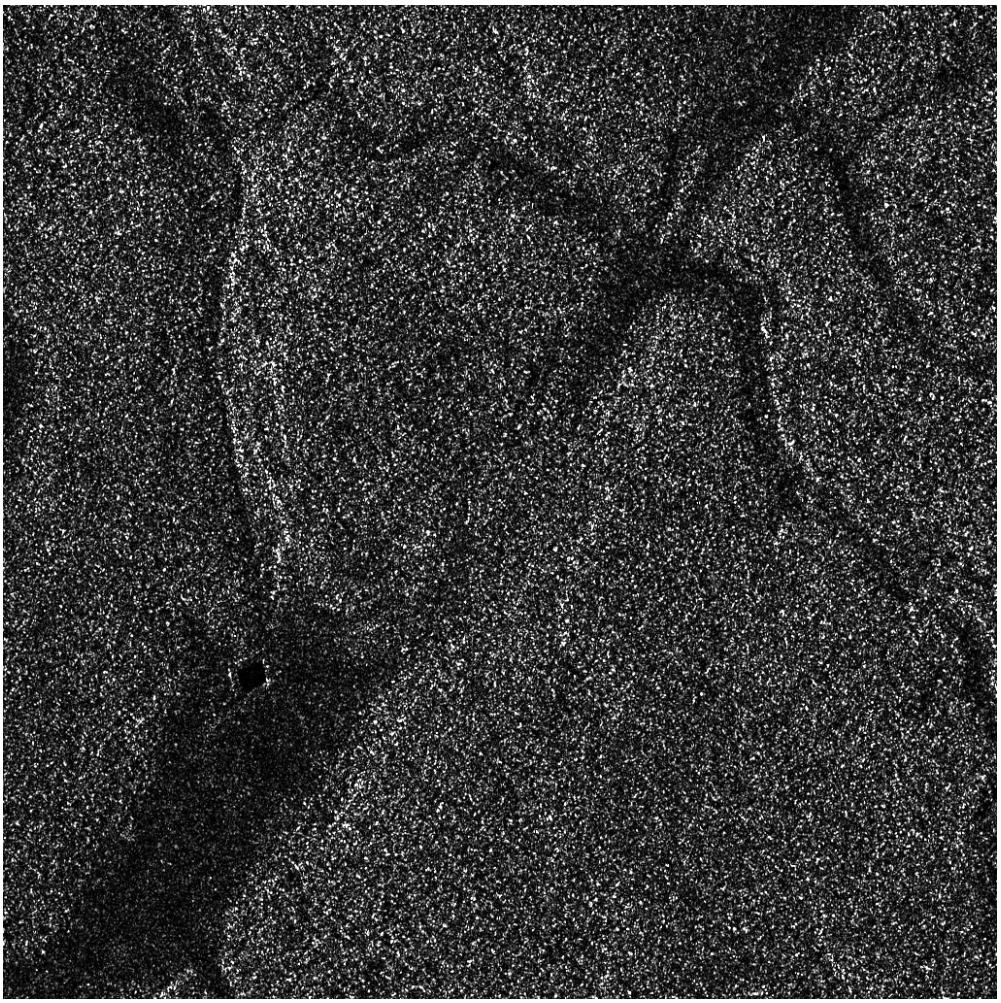
Introduction to SAR images

1. Item 1 Monteverdi detects the image as a complex data set. It provides the single-band complex image as well as 4 different points of view of the image:
   (a) the real part;
   (b) the imaginary part;
   (c) the modulus (or amplitude): $\sqrt{Re^2 + Im^2}$;
   (d) the phase: arctan $\frac{Im}{Re}$.

2. Item 2
   (a) The modulus image corresponds to the energy of the backscatter received by the sensor and is correlated to roughness, moisture and slope of the surface. The other 3 images (Re, Im and phase) do not contain visual information about the landscape. One can observe that the Re and Im parts are rather homogeneous in the shadow areas, since no signal is relected back to the sensor.
   (b) The Re and Im parts are Gaussians with a zero mean. As a consequence, the complex SAR image will have a circular Gaussian distribution, whose phase has an uniform distribution in $[-\pi, \pi]$ (Central limit theorem) and whose modulus follows a Rayleigh distribution.

3. Item 3 The image intensity is the square of the amplitude. Using the *BandMath* module, there are 2 ways of computing the amplitude:
   (a) Use the modulus image as input to *BandMath* and compute the square of the image: `im1b1*im1b1`.
   (b) Use the Re and Im parts as input and compute the sum of the squares: `im1b1*im1b1+im2b1*im2b1`.

4. Item 4 Here is the ROI:



Speckle reduction

1. Item 1 The despeckle filtering with any of the proposed values for the radius produce a major enhancement of the image and allow to identify features which were barely visible on the original image.

   Increasing the value of the radius increases the amount of smoothing of the filtering, since larger areas are taken into account. This produces increased enhancements on the homogeneous areas of the image, but introduces a loss of details and even a deformation of the shapes where strong contrasts are present.

2. Item 2 The histograms of the filtered images become increasingly Gaussian (symetric bell-shaped function) and progressively differ from the Gamma distribution (asymetric bell-shaped function with a long tail towards the right) of the original image.

3. Item 3 Higher values of the deramp parameter produce a slower decay of the exponential weighting if the Frost filter and therefore, the smoothing effect increases.

3.9 Multi temporal image analysis

3.9.1 Description

Abstract This exercise will get you familiar with multi-temporal image analysis using OTB applications. You will learn how to script OTB applications with Python to facilitate manipulation of a set of LandSat-8 images and produce change detection map from the image series.

Data We'll use Landsat 8 images. It provides moderate-resolution imagery, from 15 meters to 100 meters with a revisit time of approximately 16 days. Images were download on the USGS Earth Explorer website: `http://earthexplorer.usgs.gov/`

Products downloaded:

- LC81530752013182LGN00
- LC81530752013262LGN00
- LC81530752013326LGN00
- LC81530752014025LGN00
- LC81530752014073LGN00
- LC81530752014121LGN00
- LC81530752014169LGN00
- LC81530752014233LGN00

From:

- DATE ACQUIRED = 2013-07-01
- DATE ACQUIRED = 2013-09-19
- DATE ACQUIRED = 2013-11-22
- DATE ACQUIRED = 2014-01-25
- DATE ACQUIRED = 2014-03-14
- DATE ACQUIRED = 2014-05-01
- DATE ACQUIRED = 2014-06-18
- DATE ACQUIRED = 2014-08-21

Pre-requisites

- Basic knowledge on **OTB applications**
- Basic knowledge of Python

Achievements

- Usage of the OTB applications in Python
- Produce detection map from a image series

3.9.2 Steps

Concatenate image bands

- Landsat 8 images are stored in separate directories. Moreover each band are separated in different tif files. Develop a Python script which browse in each directory and concatenate image bands in a single vector image. We will use those vector images in the following steps of this exercise.

LandSat-8 imagery : basic visualization and band combinations Use Monteverdi2 or Ice to visualize Landsat 8 images.

- What is the color composition for "color infrared" visualization?
- What is the color composition for "Natural color?
- What is the number of the new coastal/aerosol band in Landsat 8's OLI sensor?
- What is the number of the new cirrus band in Landsat 8's OLI sensor?

Radiometric index extraction

- Develop a Python script which compute NDVI from all Landsat 8 images series.

Change detection using MAD MAD stands for Multivariate Alteration Detector and is a change detection analysis technique based on the following : find pairs of linear combination between bands from image 1 and bands from image 2 that maximize correlation, each pair being orthogonal with the others. Subtract each pairs of linear combination one from each other. This leads to a set of change maps which are invariant to linear scaling, un-correlated and can be used to perform change detection with images with different bands and number of bands for instance.

- What is the OTB applications which allows to perform MAD change detection?
- What is the command to perform MAD change detection between images from the 2013-07-01 and 2013-09-19?
- Which band of the MAD result seems of interest to extract change of fields?
- Try to perform threshold on this band to extract a detection mask.
- What is the command top Perform subtraction between NDVI images from the 2013-07-01 and 2013-09-19?
- Try also to threshold the result from the previous question.

3.9.3 Solutions

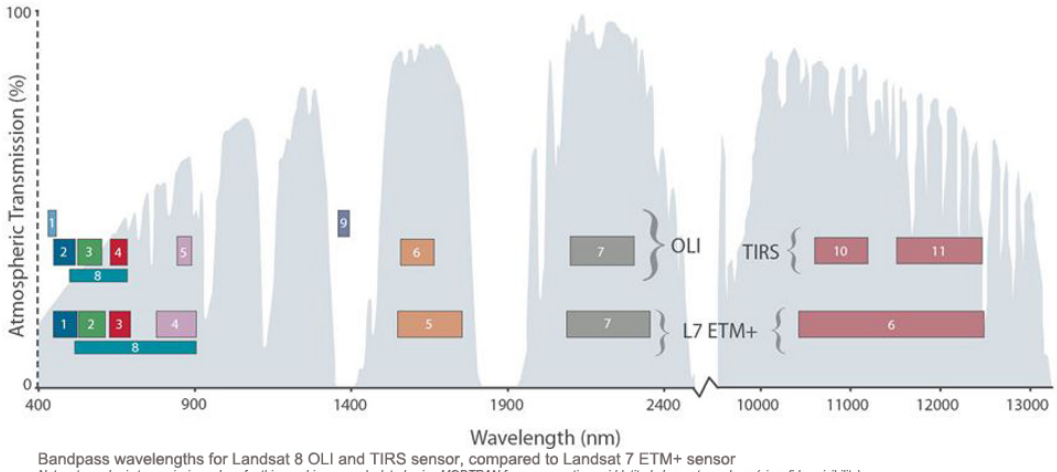
Concatenate image bands

```
#!/usr/bin/python

import sys, subprocess,re,os

def main(args):
    for dirname, dirnames, filenames in os.walk(str(args[0])):

        band_list=[]
        for i in range(1,8):
            band_list.append('B'+str(i))

        for subdirname in dirnames:
            if subdirname.find('LC8') != -1:
                print "subdir ",os.path.join(dirname, subdirname)
                img_list=[]
                process = ['otbcli_ConcatenateImages']
                process.append('-ram 512')
                process.append('-out')
                output_file=
                process.append(os.path.join(dirname, subdirname)
                               + '_concat.tif'
                               + '?&box=1540:680:3000:3000')
                #process.append('?&box=1438:613:3015:2586"')
                process.append('uint16')
                process.append('-il')
                #print process
                for band in band_list:
                    process.append(os.path.join(
                                   os.path.join(dirname, subdirname),
                                   subdirname + '_' + band + '.TIF'))
                print process

                subprocess.check_call(process)

if __name__ == '__main__':
    main(sys.argv[1:])
```

LandSat-8 imagery : basic visualization and band combinations

- Color Infrared: 5,4,3
- Natural Color: 4,3,2
- coastal/aerosol band (band1) (use clamp around min/max with 2% and 8%)

- cirrus band (band 9) : better detection of cirrus cloud contamination (problem with Reunion data)



Bandpass wavelengths for Landsat 8 OLI and TIRS sensor, compared to Landsat 7 ETM+ sensor
Note: atmospheric transmission values for this graphic were calculated using MODTRAN for a summertime mid-latitude hazy atmosphere (circa 5 km visibility).

Radiometric index extraction

```
#!/usr/bin/python

import sys, subprocess,re,os

def main():
    for dirname,dirnames,filenames in os.walk(str(args[0])):
        band_list=[]
        for i in range(1,8):
            band_list.append('B'+str(i))

        for filename in filenames:
            if re.match(r'(.*)_concat.tif($)',filename):
                print "filename ",os.path.join(dirname, filename)
                argv = ['otbcli_BandMath']
                argv.append('-il')
                argv.append(os.path.join(dirname, filename))
                argv.append('-out')
                argv.append(os.path.splitext(os.path.join(dirname,filename))[0]
                argv.append('float')
                argv.append('-exp')
                argv.append('ndvi(im1b4,im1b5)')

                subprocess.check_call(argv)
if __name__ == '__main__':
    main(sys.argv[1:])
```

Change detection using MAD

- **MultivariateAlterationDetector**

- Here is the command line to produce the MAD change detection on input image:

```
$ otbcli_MultivariateAlterationDetector
  -in1 LC81530752013182LGN00_concat.tif
  -in2 LC81530752013262LGN00_concat.tif
  -out ~/temporary/mad_182_262.tif
```

- It seems that band number 5 allows to discriminate changes on agricultural lands.

- Threshold using the **BandMath** application. I choose to threshold mad values superior to 4.5:

```
$otbcli_BandMath
-il ~/temporary/mad_182_262.tif
-out ~/temporary/mad_182_262_thres.tif uint8
-exp "im1b5>=4.5?255:0"
```

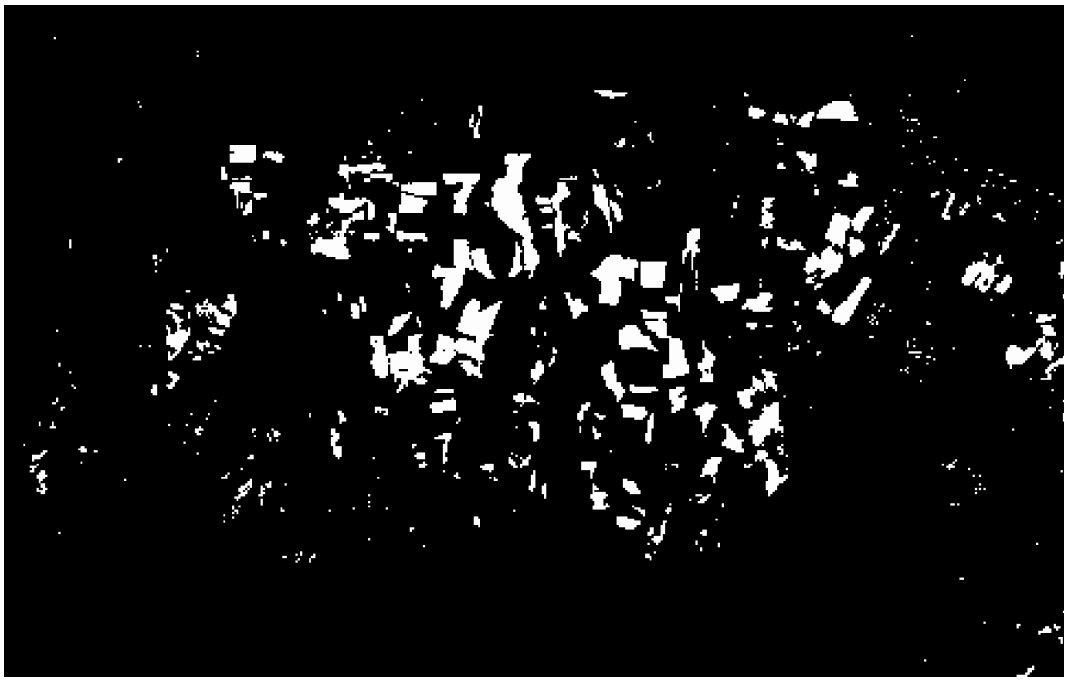
Still need to filter clouds. MAD filter could take a mask as input in the future.

Results:

Extract from the 2013-07-01:

Extract from the 2013-09-19:

Extract from the change detection result:



- TODO
- TODO

3.10 Monteverdi and OTB applications

3.10.1 Description

Abstract This exercise will get you familiar with the use of **Monteverdi** and **OTB applications**. Monteverdi is the OTB composer which allows building processing chains by selecting modules from a set of menus.It was developed in 2009 and is now replace by a brand new software call **Monteverdi2**.

If you want to learn more about **Monteverdi2** you can see `this exercise`.

Data If you need to generate the data used in this exercise from the original products, you can use the following command lines.

```
$ otbcli_ExtractROI \
 -in ORTHO_UTM_PMS/IMG_PHR1A_PMS_001/IMG_PHR1A_PMS_201202250025599_ORT_IPU_20120504_1772-001_R1C1.JP2 \
 -out phr_pxs_melbourne.tif uint16 -startx 18432 -starty 4096 -sizex 4096 -sizey 4096

$ otbcli_ExtractROI \
  -in PRIMARY_TRISTEREO_BUNDLE/IMG_PHR1A_MS_006/IMG_PHR1A_MS_201202250025599_SEN_IPU_20120509_2001-012_R1C1.JP2 \
  -out phr_xs_melbourne.tif uint16 -startx 4096 -starty 2048 -sizex 4096 -sizey 4084
```

Pre-requisites

- Basic knowledge of remote sensing and image processing,
- Basic knowledge of command-line invocation.

Achievements

- Visualize data in **Monteverdi**,
- Basic processing in **Monteverdi**,
- Basic processing with **OTB applications** in graphical mode,
- Basic processing with **OTB applications** in command-line mode.

3.10.2 Steps

Monteverdi: data opening and visualisation In this part of the exercise, you will use the following data: `phr_pxs_melbourne.tif`

1. Run **Monteverdi**: open a terminal and run the following command:

   ```
   $ monteverdi
   ```

2. Open the image (use the *File/Open dataset* menu)
3. Find how to display the image (there are two ways of doing it).
4. Navigate into the image:
   (a) Change the full resolution displayed area
   (b) Change the zoom displayed area,
   (c) Change the zoom level,
   (d) What are the information displayed about the current pixel under mouse pointer ?
5. Using the viewer control panel, in the *Setup* tab:
   (a) Change set-up to visualize the 4th band,
   (b) Change set-up to visualize in false color mode (screen red: near infra-red channel, screen green: red channel, screen blue: green channel).
   (c) Change set-up to come back to natural colors
   (d) Enhance contrast with respect to the full area,
   (e) Enhance contrast with respect to the zoom area,
   (f) Come back to default contrast enhancement.

Tips and Recommandations:

- You can use keyboard arrows to navigate into images as well,
- Pleiades bands order is red channel, green channel, blue channel, near infra-red channel.

Monteverdi: basic processing In this part of the exercise, you will use the following data:
`phr_xs_melbourne.tif`

1. Open the image in **Monteverdi**.
2. Find the *BandMath* module in the menu. Open the image in this module. What kind of processing is offered ?
3. Using this module, compute the NDVI of the image:

$$NDVI = \frac{NIR - RED}{NIR + RED} \tag{2}$$

Visualize the input image and the mask in the same viewer.

1. Using this module, build a mask of pixels whose Digital Number (DN) in the NIR channel is lower than 150. Visualize the input image and the mask in the same viewer.
2. Using this module, build a mask of pixels whose DN is upper than 1000 in all spectral bands. Visualize the input image and the mask in the same viewer.
3. Using the *Concatenate* module, build a composite RGB image with the mask of high values in the red channel, the mask of low NIR values in the blue channel and the NDVI in the green channel.
4. Using the *Color Mapping* Module, build a composite RGB image of the NDVI that allows for better image interpretation.

Tips and Recommandations:

- NDVI values are within -1 and 1, but the range can be much more narrow.

OTB applications: Graphical and command-line mode

1. Run the following command:

```
$ otbcli_OrthoRectification
```

And then

```
$ otbgui_OrthoRectification
```

What do you observe ?

2. How many **OTB applications** are currently available ?
3. How can you get help and documentation about applications ?

OTB applications: Basic processing

1. Use the **OTB applications** to produce the same results as steps 3 to 7 as with **Monteverdi** in section 3.10.2.

Homework

1. How can we load or visualize images directly from command-line using **Monteverdi** ?
2. Is there another way to compute radiometric indices like NDVI with the **OTB Applications** ?
3. Learn about the *Python* access to **OTB Applications** and write a python script performing the same steps as in section 3.10.2.

3.10.3 Solutions

Monteverdi: data opening and visualisation

1. Item 3

   To load an image into **Monteverdi** viewer module, you can either:

   - Right-click on the image and select *Display in viewer*,
   - In the menu bar, select *Visualization/Viewer*, select the corresponding image and push *Ok*.

   The latter allows to load multiple images into a single viewer.

2. Item 4

   The lower left text area displays information on the image and on the pixel under the mouse pointer:

   - The current position in image,
   - The image size,
   - The channel displayed,
   - The pixel values,
   - The estimated ground spacing,
   - The geographic position (if available),
   - The current location (if available).

Monteverdi: basic processing

1. Item 2

   The **BandMath** module allows to do advanced band calculations using the syntax from muParser .

2. Item 3

   To compute the NDVI, use the following **BandMath** expression:

   ```
   (im1b4-im1b1)/(im1b4+im1b1)
   ```

3. Item 4

   To build a mask of pixels whose DN in the NIR channel is lower than 150, use the following **BandMath** expression:

```
if(im1b4<150,255,0)
```

4. Item 5

   To build a mask of pixels whose DN is upper than 1000 in all spectral bands, use the following **BandMath** expression:

   ```
   if(min(im1b1,im1b2,im1b3,im1b4)>1000,255,0)
   ```

5. Item 6

   In the menu bar, select *File/Concatenate images*, and loads the three **BandMath** module outputs. The resulting image can be displayed in the viewer and will look like this:

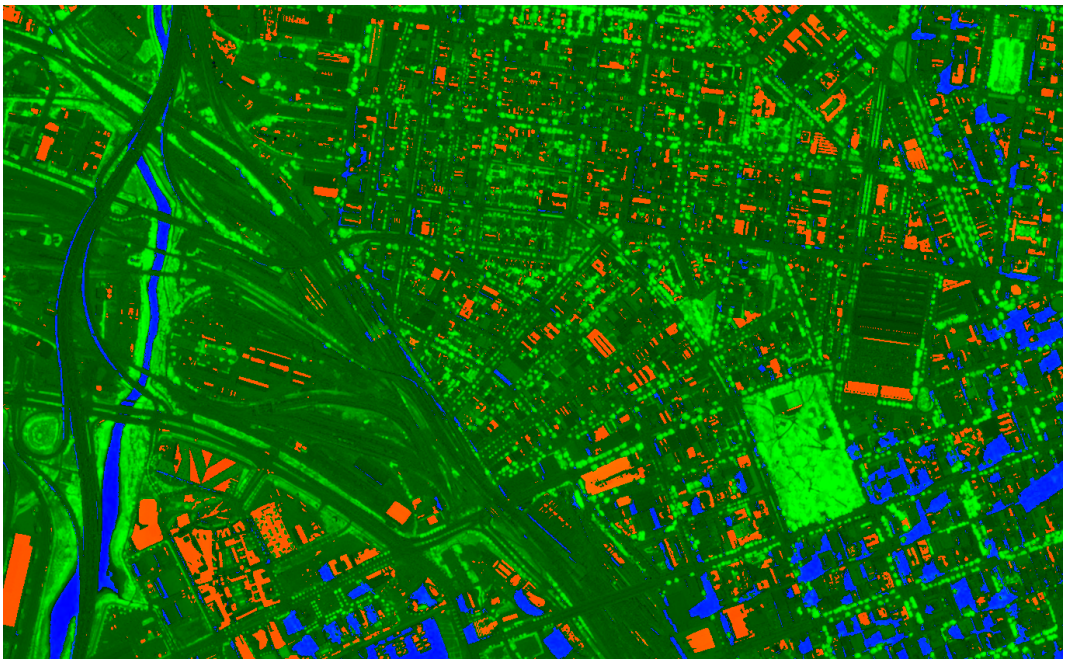


Figure 5: Results of the concatenation of Band Math

6. Item 7

   In the menu bar, select *Visualisation/Color Mapping* and load the NDVI output from the **BandMath** module. Set a mapping range from -0.2 to 0.7 so as to adapt to NDVI range, and select the *Jet* color map. The resulting image can be displayed in the viewer and will look like this:

OTB applications: Graphical and command-line mode

1. Item 1

   The first command runs the command-line version of the **Orthorectification** application, the second one runs the graphical version.

2. Item 2

   There are 59 applications available in OTB 3.14.1.

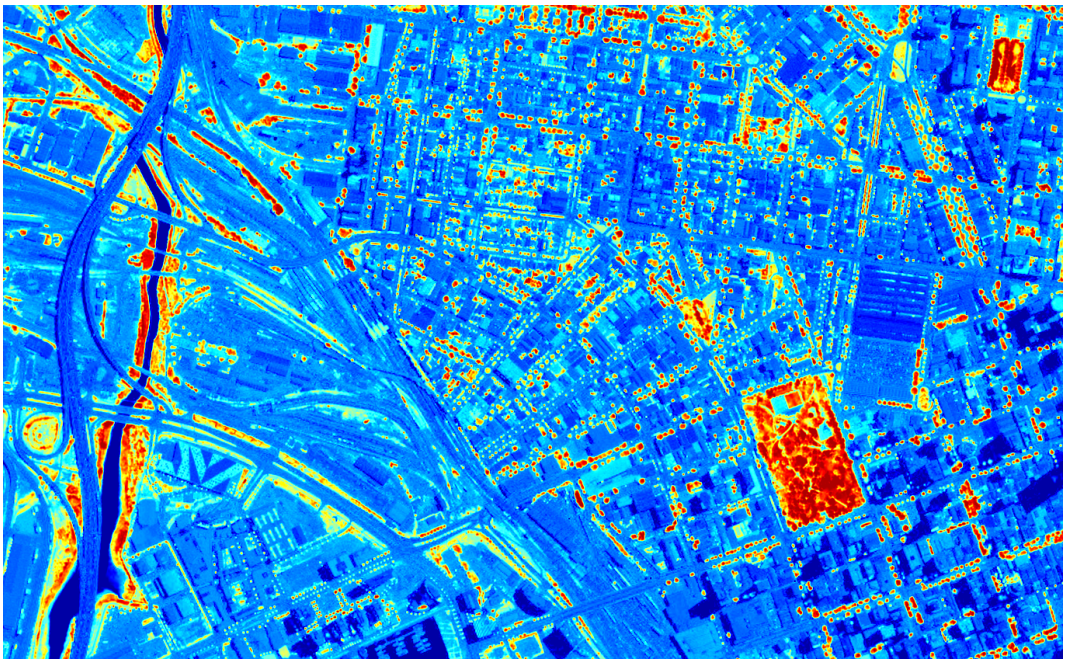


Figure 6: Color mapping of NDVI using Jet color map

3. Item 3

   There are several ways to get help and documentation:

   - Running the command-line version of the application displays a short description of the parameters, and also gives a link to the documentation on the OTB website,
   - Running the graphical version of the application shows a *Documentation* tab where extensive documentation of parameters can be found.
   - Last, the complete applications documentation can be found in the Orfeo ToolBox Cookbook.

OTB applications: Basic processing

1. Item 1

   Here is the set of commands to reproduce the processing from section 3.10.3.

   First, we compute the NDVI with the **BandMath** application:

```
$ otbcli_BandMath -il phr_xs_melbourne.tif
  -out ndvi.tif float -exp "(im1b4-im1b1)/(im1b4+im1b1)"
```

   Then, we compute the mask of pixels whose DN in the NIR channel is lower than 150:

```
$ otbcli_BandMath -il phr_xs_melbourne.tif
  -out lownir.tif uint8 -exp "if(im1b4<150,255,0)"
```

Next, we compute the mask of pixels whose DN is upper than 1000 in all spectral bands:

```
$ otbcli_BandMath -il phr_xs_melbourne.tif
  -out high.tif uint8
  -exp "if(min(im1b1,im1b2,im1b3,im1b4)>1000,255,0)"
```

Please note that for masks using a *uint8* data type is enough, while for NDVI a floating point data type is needed.

Now, we can concatenate all outputs in a single map with the **ConcatenateImages** application:

```
$ otbcli_ConcatenateImages -il high.tif ndvi.tif lownir.tif
  -out map1.tif float
```

Finally, we can create a color-mapping of the NDVI using the **ColorMapping** application:

```
$ otbcli_ColorMapping -in ndvi.tif -out map2.png uint8
  -method continuous -method.continuous.min -0.2
  -method.continuous.max 0.7 -method.continuous.lut jet
```

Homework

1. Item 1 From the command-line, running

   ```
   $ monteverdi -in  phr_xs_melbourne.tif
   ```

   will open the image in **Monteverdi** and display it in the viewer, and

   ```
   $ monteverdi -il  phr_xs_melbourne.tif ndvi.tif
   ```

   allows to open a list of images in **Monteverdi**.

2. Item 2 In **OTB Applications**, there is a **RadiometricVegetationIndices** application that allows to compute several indices including the NDVI.

3. Item 3

   Please refer to this chapter of the **Cookbook** to learn more about the *Python* interface.