Orfeo ToolBox

www.orfeo-toolbox.org

CENTRE NATIONAL D'ÉTUDES SPATIALES

OPEN-SOURCE REMOTE SENSING

What is Orfeo ToolBox ?

Orfeo ToolBox in a nutshell

- It is an image processing library dedicated to remote sensing
- It is an open-source software under CeCill-v2 license (french equivalent to GPL)
- It is funded and developed by CNES in the frame of the ORFEO acompaniement program
- It is written in C++ on top of ITK (medical image processing)
- It interfaces seamlessly with other image processing and remote sensing open-source softwares, like GDAL or OSSIM
- Available on multiple platforms (Windows, Linux, Mac OS X)
- It is scalable thanks to parallel and on the flow processing

Thanks to its modular architecture, Orfeo ToolBox allows fast prototyping and development of processing-chains.

What you can do with Orfeo ToolBox

- Read, write, convert, extract parts of your remote sensing data,
- Perform basic pre-processing like ortho-rectification, radiometric calibration or pan-sharpening,
- Perform common image processing tasks (thresholding, dimensionality reduction, Fourier or wavelets transform . . .)
- Extract features (radiometric indices, textures, shapes . . .),
- Segment images and vectorize segmentation results,
- Classify images in a supervised or unsupervised way,
- Perform object-based image analysis,
- Export results in Google Earth, Qgis and pretty print for publishing.

Pleiades image in Orfeo ToolBox

- Supports Jpeg2000 image format through the open-source library OpenJPEG
- Supports projection information and analytical sensor models through OSSIM
- Supports calibration information to convert DN reflectance values to absolute radiance values
- Decoding of intermediate resolutions
- Efficient viewing and navigation with Monteverdi (see below)

Getting help and support

- As an open-source software, OTB has its own users and developers community
- The development team provides support through mailing-list (otb-users@googlegroups.com) or IRC channel
- Features request can also be sent this way
- Want to keep in touch with the OTB ? Bookmark our blog (`blog.orfeo-toolbox.org`)

How can I use Orfeo ToolBox ?

By programming

Know a little about programming and want to access the full range of Orfeo ToolBox functions to build custom tools ? Dive into our Software Guide and start writing your own C++ code with OTB.

By running OTB applications

Want to benefit from the power of full OTB processing chains from your favorite software environment ? Try our application plugins, shipped with a command-line interface, a QT graphical interface and a python (and other high-level languages) one for remote sensing tasks scripting.

By using Monteverdi

Want an integrated software for everyday life image manipulation or support for your training courses ? Try Monteverdi, our end-user software featuring a nice image viewer and a range of image processing modules based on OTB.

What does Orfeo ToolBox look like ?

A simple example of C++ OTB code

```cpp
#include "otbImage.h"
#include "otbImageFileReader.h"
#include "otbStreamingImageFileWriter.h"
#include "itkCannyEdgeDetectionImageFilter.h"
#include "itkRescaleIntensityImageFilter.h"

int main(int argc, char * argv[])
{
  typedef double                      PixelType;
  typedef otb::Image<PixelType>       ImageType;

  typedef unsigned char               OutputPixelType;
  typedef otb::Image<OutputPixelType> OutputImageType;

  typedef otb::ImageFileReader<ImageType> ReaderType;
  ReaderType::Pointer reader = ReaderType::New();

  reader->SetFileName(argv[1]);

  typedef itk::CannyEdgeDetectionImageFilter
  <ImageType, ImageType> FilterType;
  FilterType::Pointer filter = FilterType::New();

  filter->SetInput(reader->GetOutput());

  typedef itk::RescaleIntensityImageFilter
  <ImageType, OutputImageType> RescalerType;
  RescalerType::Pointer rescaler = RescalerType::New();

  rescaler->SetOutputMinimum(0);
  rescaler->SetOutputMaximum(255);

  rescaler->SetInput(filter->GetOutput());

  typedef otb::StreamingImageFileWriter<OutputImageType> WriterType;
  WriterType::Pointer writer = WriterType::New();

  writer->SetFileName(argv[2]);

  writer->SetInput(rescaler->GetOutput());

  writer->Update();

  return EXIT_SUCCESS;
}
```

How can I help ?

We do not only need developers ! To get involved, you can :

- Help us to validate algorithms and send feedback
- Provide OTB case studies (user stories)
- Help us to track bugs on `bugs.orfeo-toolbox.org`
- Provide new ideas and feature requests
- Spread the word !

Calling applications from command-line

```
$ otbcli_ImageSVMClassifier -in QB_1_ortho.tif -imstat
  clImageStatisticsQB1.xml -svm clsvmModelQB1.svm -out
  classification.png uchar
```

Calling applications from Qt interface

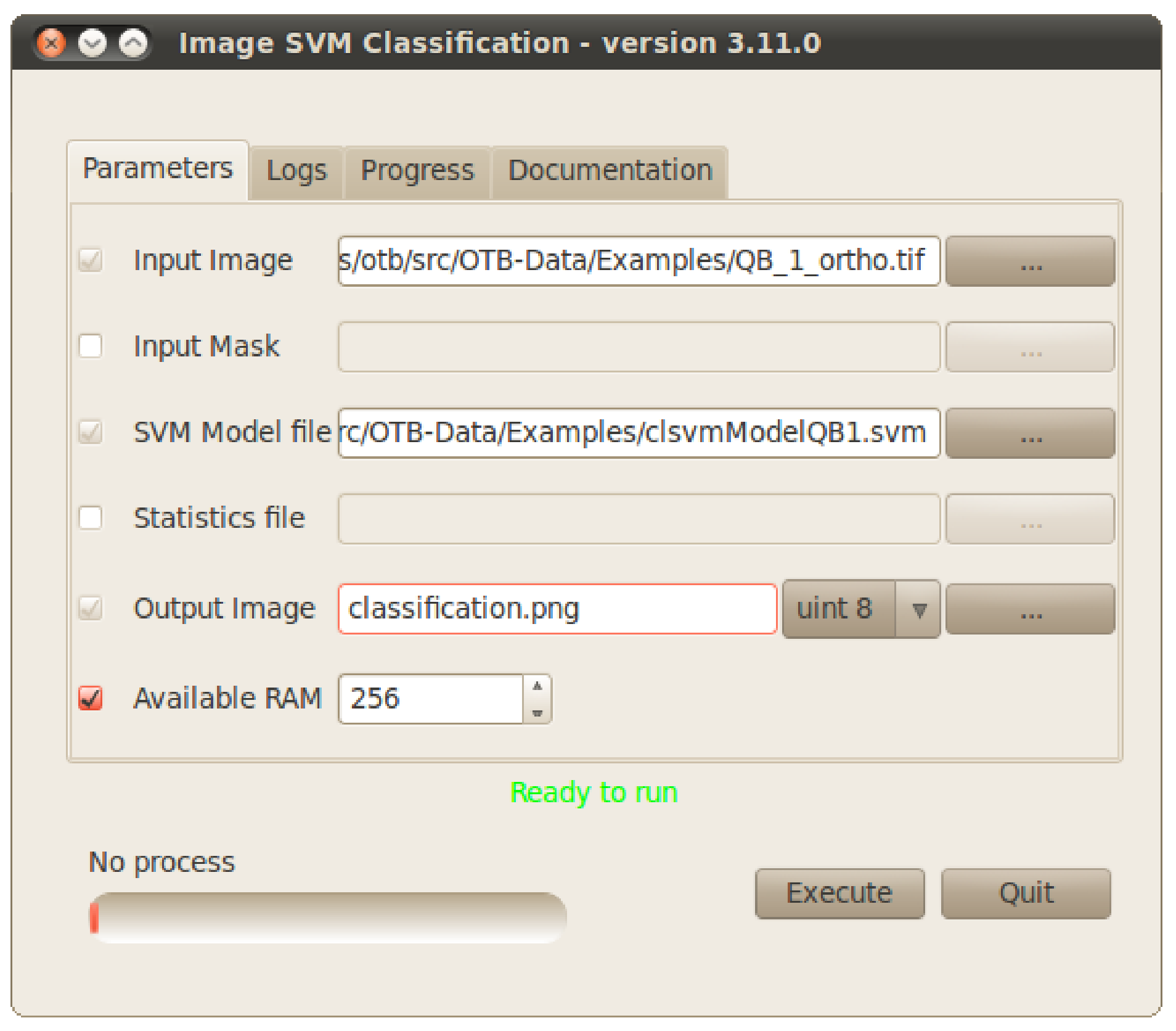


Image SVM Classification - version 3.11.0

Parameters | Logs | Progress | Documentation

- [✓] Input Image: s/otb/src/OTB-Data/Examples/QB_1_ortho.tif
- [ ] Input Mask:
- [✓] SVM Model file: rc/OTB-Data/Examples/clsvmModelQB1.svm
- [ ] Statistics file:
- [✓] Output Image: classification.png — uint 8
- [✓] Available RAM: 256

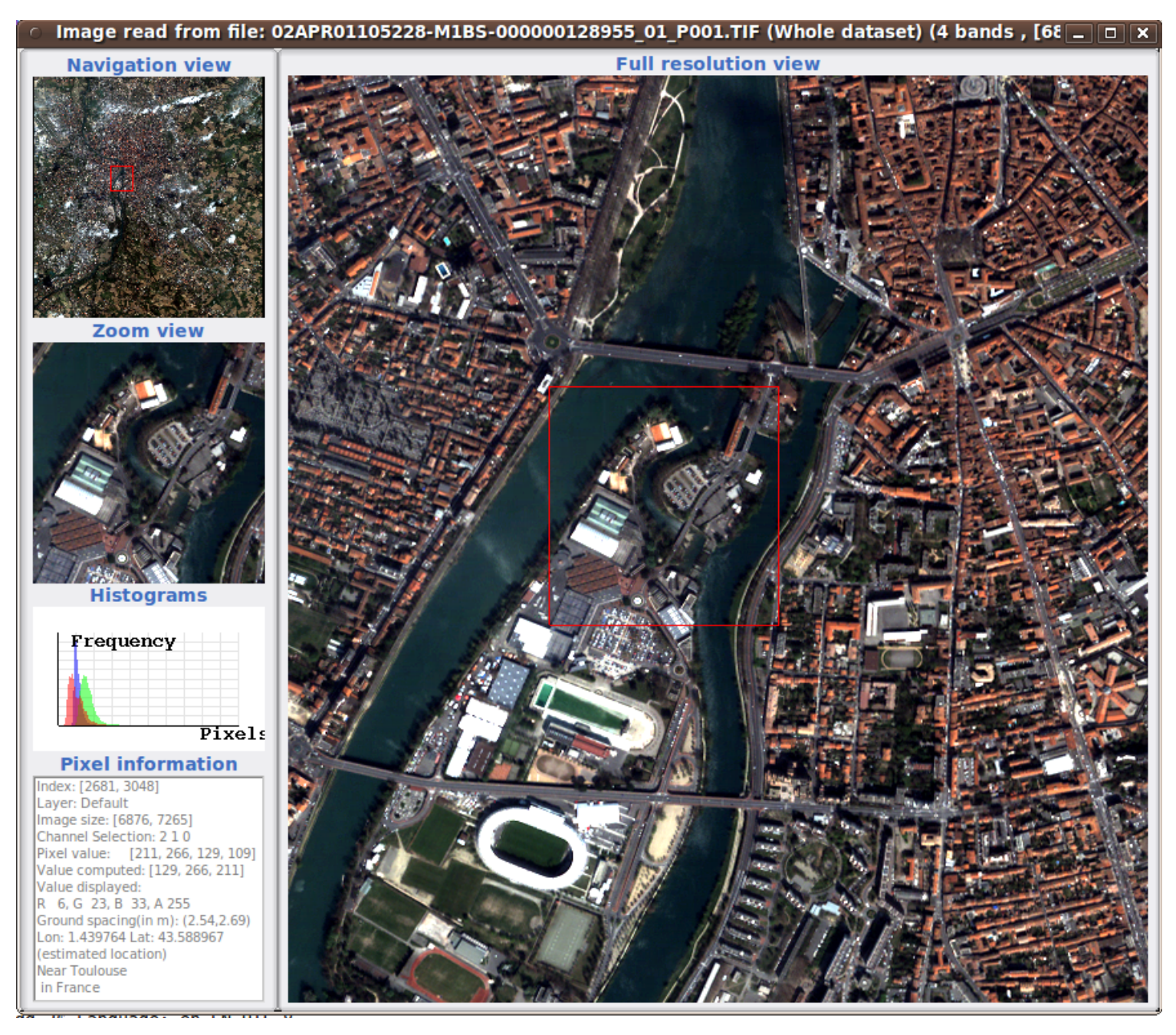
Ready to run

No process

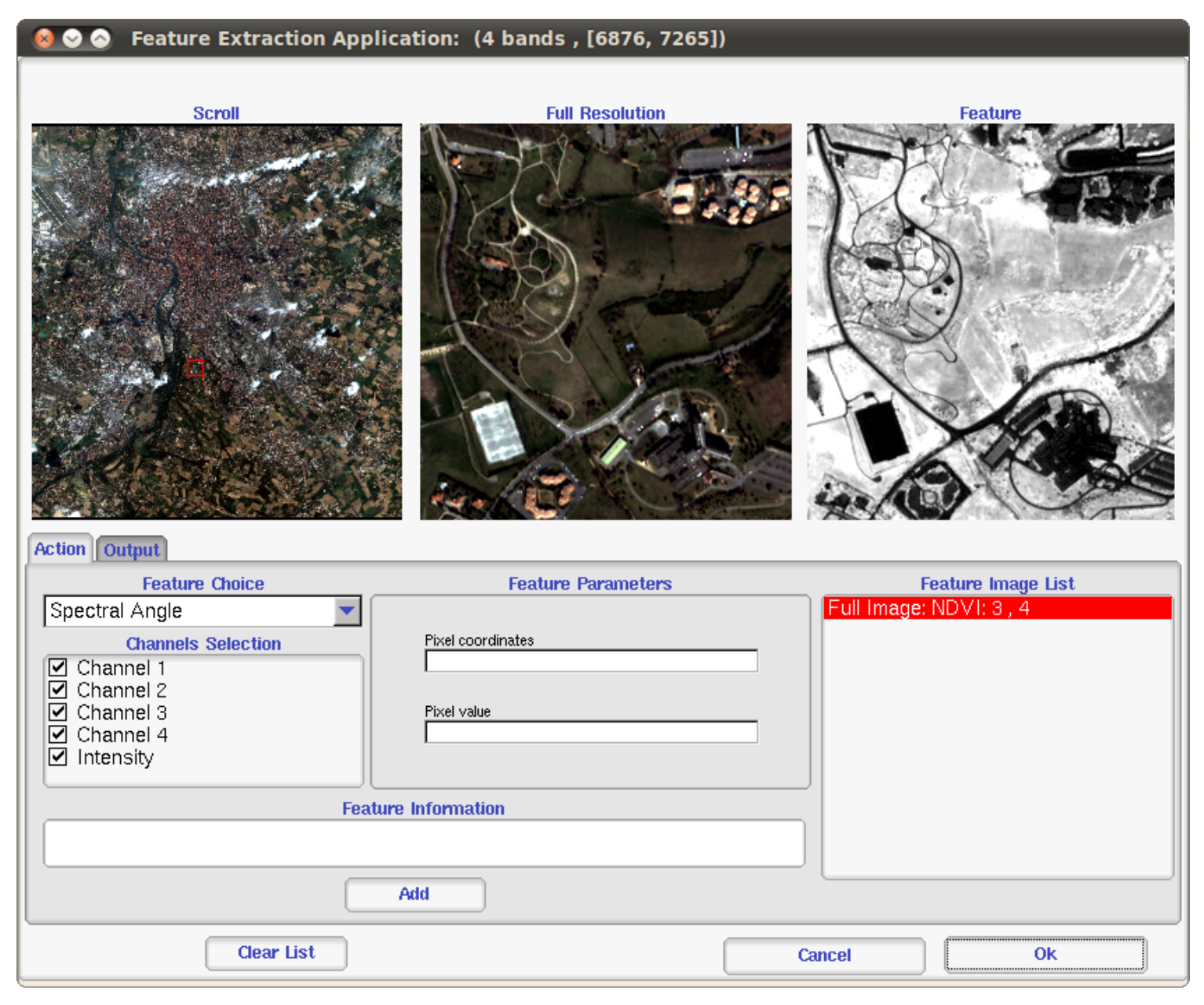
Calling applications from python

```python
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the ImageSVMClassifier
  application
ImageSVMClassifier =
  otbApplication.Registry.CreateApplication("ImageSVMClassifier")

# The following lines set all the application parameters:
ImageSVMClassifier.SetParameterString("in", "QB_1_ortho.tif")
ImageSVMClassifier.SetParameterString("imstat",
  "clImageStatisticsQB1.xml")
ImageSVMClassifier.SetParameterString("svm", "clsvmModelQB1.svm")
ImageSVMClassifier.SetParameterString("out", "classification.png")
ImageSVMClassifier.SetParameterOutputImagePixelType("out", 1)

# The following line execute the application
ImageSVMClassifier.ExecuteAndWriteOutput()
```

Monteverdi viewer



Monteverdi feature extraction module



Monteverdi segmentation module