Guide logiciel de l'OTB
Première édition
Version DRAFT

J. Inglada (*CNES*)
T. Feuvrier (*CS*)

December 16, 2005

http://www.cnes.fr
Email: jordi.inglada@cnes.fr
thomas.feuvrier@c-s.fr

Résumé

Parallèlement aux développements des systèmes *Pléiades*[1] (PHR) et *Cosmo-Shymed* constituant le système dual et bilatéral (France-Italie) d'observation de la Terre *ORFEO*, le programme d'accompagnement d'ORFEO a été mis en place par le CNES afin de préparer, accompagner et promouvoir l'utilisation et l'exploitation des images issues de ces capteurs.

Le volet méthodologique de ce programme d'accompagnement[2] a pour objectif la définition et le développement des outils nécessaires à l'exploitation opérationnelle des futurs images submétriques, optique et radar (aspects tridimensionnels, détection de changements, analyse de texture, reconnaissance de formes, complémentarité optique-radar). Il s'appuie essentiellement sur les études de R&D et des travaux de recherche doctorale et post-doctorale.

Dans ce contexte, le CNES[3] a décidé de développer l'*ORFEO ToolBox* (OTB), un ensemble de briques algorithmiques qui permettront de capitaliser le savoir méthodologique et de se placer dans une démarche de développement incrémental visant à rentabiliser au maximum les résultats obtenus dans ces études méthodologiques.

Tous les développements de l'OTB sont basés sur des bibliothèques de type libres (licences GNU/GPL, etc..) ou des bibliothèques existantes développées par le CNES.

OTB est implémentée en C++, dont le développement s'appuie principalement sur la bibliothèque ITK[4] (Insight Toolkit) et sur les bibliothèques VTK[5] (Visualization ToolKit) FLTK[6] (Fast Light Toolkit).

Les bibliothèques GDAL[7] (Geospatial Data Abstraction Library) et CAI (Couche d'Accès Images développée par le CNES) sont utilisées comme progiciels pour la lecture et l'écriture des

[1] http://smsc.cnes.fr/PLEIADES/Fr
[2] http://smsc.cnes.fr/PLEIADES/Fr/A_prog_accomp.htm
[3] http://www.cnes.fr
[4] http://www.itk.org
[5] http://www.vtk.org
[6] http://www.fltk.org
[7] http://www.remotesensing.org/gdal/

images de télédétection.

L'environnement de l'OTB est mis en place par l'outil CMake[8], permettant ainsi de gérer les procédures de compilation, génération et d'installation et ce quelque sois la plate forme cible.

Dans un souci d'homogénéisation, l'OTB est conçue et développée suivant la philosophie et les principes édictés par la bibliothèque ITK (programmation générique, mécanisme des *Object Factories*, *Smart pointers*, exceptions, *Multi-Threading*, etc...). Ces principes sont présentés dans le paragraphe *3.2 Essential System Concepts* du guide ITK `http://www.itk.org/ItkSoftwareGuide.pdf`

Enfin, la méthodologie de développement appliquée s'appuie sur une approche itérative basée sur la programmation agile : le schéma de développement suit le cycle édictée par la méthodolgie de l'eXtreme Programming (XP)[9].

Ce document constitue le guide d'utilisation et de développement de l'OTB. La version la plus récente de ce document est accessible à `http://smsc.cnes.fr/PLEIADES/Fr/A_prog_accomp.htm/OTB/otbSoftwareGuide.pdf`.

[8] http://www.cmake.org
[9] http://www.xprogramming.com

Contributeurs

L'ORFEO ToolBox (OTB) est un projet mis en place par le CNES et développé par la société "Communication & Systèmes" (CS)

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Part I

Introduction

CHAPTER

ONE

Bienvenue

Bienvenue dans le l'*guide de l'ORFEO ToolBox (OTB)*.

Ce document vous présente les concepts importants utilisés dans l'OTB et vous guide efficacement dans son apprentissage et son utilisation. De plus, la documentation au format Doxygen du code source de l'OTB est également accessible.

1.1 Organisation

Ce guide logiciel est divisé en trois parties, chacune étant divisée en chapitres.

La première partie présente l'OTB de façon générale, comment procéder à son installation et sa génération sur votre machine. Cette partie présente donc les principes de bases d'architecture et de compilation sur un sytème, et comment compiler une application en C++.

La deuxième partie présente l'OTB d'un point de vue *utilisateur*. Elle se présente sous forme d'exemples illutrés.

La troisième partie présente l'OTB d'un point de vue *développeur*. Cette partie explique comment créer vos propres classes, comment faire évoluer le produit.

1.2 Se familiariser avec l'OTB

Il y à deux catégories d'utilisateurs de l'OTB :

- Les développeurs de classes, qui créent des classes C++.
- Les utilisateurs des classes existantes pour développer et générer leurs propres applications.

Nous vous recommendons d'étudier les exemples. Vous pourrez ainsi compiler et exécuter les exemples distribués avec le code source diponible dans le répertoire `OTB/Examples`. Lire le fichier `OTB/Examples/README.txt` décrivant les différents exemples founis dans les sous-répertoires.

Il y à de plus, un ensemble de tests suffisamment documentés et disponibles dans le répertoire `OTB/Testing/Code` qui vous montrent comment peuvent être utilisées les classes dans l'OTB.

1.3 Organisation du logiciel

En cours ....

1.4 Télécharger l'OTB

L'OTB peut être téléchargée à l'adresse Internet

`http://www.cnes.fr/HTML/Download.php`

1.4.1 Télécharger le 'Package'

Avant de démarrer, vous pouvez consulter le document `GettingStarted.txt`[1]. Il vous donne un aperçu sur la procédure à suivre pour le téléchargement et l'installation.

Choisir le fichier compressé `.zip` ou `.tgz`. Le premier est plutôt destiné pour l'environnement *Microsoft Windows*, le second pour les environnements *unix* ou *linux*.

Extraire le contenu du fichier compressé (avec *unzip* ou *gunzip*) dans le répertoire `OTB` préalablement créé sur votre sytème. Vous êtes alors prêt à procéder à la configuration et l'installation du produit, décrite au chapitre 2.1.1 à la page 8.

1.4.2 Télécharger depuis SVN

Le code source de l'OTB est accessible via un serveur Subversion SVN (remplaçant du célèbre CVS)

Pour accéder à l'OTB via SVN (sous UNIX et Cygwin), utilisez la commande suivante :

```
svn ......
```

[1] http://www.cnes.fr/HTML/GettingStarted.txt

Ceci permet de télécharger le répertoire `OTB` contenant l'ensemble du code source de la bibliothèque *OTB*.

Vous pouvez ensuite configurer et installer l'OTB sur votre système en suivant les instructions décrites au chapitre 2.1.1 à la page 8)

1.4.3 Arborescence du produit

L'OTB est organisé en trois principaux composants : la bibliothèque OTB (répertoire `OTB`), les applications de l'OTB (répertoire `OTB-Applications`) et la documentation associée (répertoire `OTB-Documents`).

Le code source ainsi que les exemples se trouvent dans le répertoire `OTB`; la documentation, le tutorial et les procédures d'installation se trouvent dans le répertoire `OTB-Documents` ; les applications plus complexes (de plus haut niveau) se trouvent dans le répertoire `OTB-Applications`.

L'`OTB` contient les répertoires suivants :

- `OTB/Code` — contient globalement l'ensemble du code source de la bibliothèque OTB
- `OTB/Documentation` — contient la documentation de la bibliothèque OTB, produite par Doxygen
- `OTB/Examples` — contient un ensemble d'exemples, utilisés notamment pour présenter le concept de l'OTB et également utilisé pour illustrer le guide de l'OTB
- `OTB/Testing` — contient un certain nombre de programmes, utilisés pour tester et valider la bibliothèque OTB. Ces tests sont lancés via le moniteur de test de CMake *ctest*.
- `OTB/Utils` — contient les codes sources des bibliothèques utilisées par l'OTB

Le répertoire `OTB/Code` (le coeur du logiciel) est structuré de la façon suivante :

- `OTB/Code/Common` — définitions de macro, typedefs, et toutes autres classes "factorisées" utilisées par les autres composants de l'OTB.
- `OTB/Code/IO` — classes d'entrées/sorties pour l'accès aux images (encapsulation de GDAL et de CAI)
- `OTB/Code/ChangeDetection` — les classes de détections de changements
- `OTB/Code/FiterExtraction` — les classes contenant les primitives et descripteurs implémentés
- `OTB/Code/Learning` — les classes d'apprentissage supervisé (utilisant SVM)
- `OTB/Code/Visu` — les classes de visualisation et des IHM graphiques (utilisant VTK et FLTK)

Le répertoire `OTB-Documents` contient les répertoires suivants :

- `OTB-Documents/Latex` — fichiers LaTeX utilisés pour la production de documents.
- `OTB-Documents/SoftwareGuide` — fichiers LaTeX utilisés pour générer ce guide. Les exemples illustrés dans ce guide sont générés à partir des codes sources, contenus dans le rpertoire `OTB/Examples`, traduits en LaTeX

La documentation `OTB-Documents` est disponible via SVN en utilisant la commande :

```
svn ....
```

Le rpertoire `OTB-Applications` contient les répertoires suivants :

- `OTB-Applications/Chgts` — application intéractive de détection de changements, utilisant l'`OTB`
- `OTB-Applications/Viewer` — outil de visualisation d'images
- `OTB-Applications/Utils` — contient divers utilitaires comme par exemple un générateur de quick-looks, un outil d'extraction de régions d'intérêt, un outil d'affichage des méta-données des images et un outil de pseudo-ortho-rectification automatique d'images

Pour accéder aux applications de l'OTB via SVN (sous UNIX et Cygwin), utilisez la commande suivante :

```
svn ......
```

1.4.4 Documentation

Associée à ce document, il existe deux autres documentations :

La Documentation Doxygen . La documentation Doxygen est une documentation essentielle pour développer avec l'OTB. Sous format HTML, elle décrit en détail chaque classes et méthodes implémentées dans l'OTB. Elle est illustrée par des diagrammes de collaboration et des diagrammes d'héritage. Cette documentation très dynamique possède des hyper-liens sur les autres autres classes et sur le code source. Cette docmentation est disponible à l'adresse `http://smsc.cnes.fr/PLEIADES/Fr/A_prog_accomp.htm/`.

Les fichiers *Header*. Chaque classe de l'OTB est implémentée dans un fichier .h et dans un fichier .cxx/.txx (.txx pour les classes gnriques (*template*))

1.4.5 Data

Les images utilisées dans ce guide proviennent de .....

CHAPTER

TWO

Installation

Ce chapitre décrit les procédures pour installer l'OTB sur votre système. Vous devez avant tout compiler l'OTB pour pouvoir l'utiliser et créer vos propres applications.

L'OTB s'appuie sur un ensemble de bibliothèques externes de type libre (licence GNU/GPL, etc...) qu'il est nécessaire de disposer sur votre système. Ces bibliothèques sont les suivantes :

- ITK[1] (*Insight Toolkit*) utilisée comme base de développement de l'OTB, pour tous les traitements de filtrage, recalage, segmentation, etc...
- VTK[2] (*Visualization ToolKit*) utilisée pour la visualisation des données
- FLTK[3](*Fast Light Toolkit*) utilisée pour la ralisation d'IHM graphiques
- GDAL[4] (*Geospatial Data Abstraction Library*) utilisée pour toutes les fonctionnailtés de lecture et d'écriture des images de télédétection
- CAI (*Couche d'Accès Images*) utilisée pour la lecture et l'écriture des images non supportées par GDAL, en particulier pour les formats d'images SPOT.
- GSL[5] (*GNU Scientific Library*) utilisée pour toutes les fonctionnalités mathématiques très spécifiques et non fournies par ITK
- SVM[6] (*Support Vector Machines*) utilisée pour la mise en oeuvre des outils d'apprentissage par *SVM*

Elles peuvent être téléchargées sur les sites référencés ci-dessus par les liens.

[1] http://www.itk.org
[2] http://www.vtk.org
[3] http://www.fltk.org
[4] http://www.remotesensing.org/gdal/
[5] http://www.gnu.org/software/gsl/
[6] http://www.csie.ntu.edu.tw/ cjlin/libsvm/

L'OTB a été développée et testée sur plusieurs types de plate-formes opérationnelles telles que *Microsoft Windows*, *Linux* (machine compatible *Intel*), *Solaris*, and *Cygwin*. Il est ainsi possible d'utiliser les compilateurs suivants :

- Visual Studio 6
- GCC 2.95.x, 2.96, 3.x

2.1 Configurer l'OTB

L'environnement de l'OTB est mis en place via l'outil CMake[7], permettant ainsi de gérer les procédures de compilation, génération et d'installation de systèmes, et ce quelque sois la plate forme cible.

CMake permet de générer les *Makefiles* pour les sytèmes UNIX, Cygwin et les *Workspaces* pour l'environnement Visual Studio de Microsoft.

CMake utilise les informations contenues dans les fichiers nommées `CMakeLists.txt` présents dans chaque répertoire de l'OTB. L'utilisateur décrit dans ces fichiers l'information nécessaire pour que CMake puisse configurer son système (recherche de librairies externes déjà installées sur votre système, etc...)

2.1.1 Préparer CMake

CMake peut être téléchargé depuis le site

`http://www.cmake.org`

La version 2.0 de CMake est la version minimale requise pour configurer l'OTB. Vous pouvez télécharger les versions "binaires" de la plupart des plates-formes comme Windows, Solaris, IRIX, HP, Mac et Linux.

Il est aussi possible de télécharger les sources de CMake sur votre système et de re-générer l'application. Les instructions sont disponibles sur le site Internet de CMake `http://www.cmake.org`.

Pour exécuter CMake, il est nécessaire de définir : le répertoire des sources (OTB_SOURCE_DIR), et le répertoire où sont produits les fichiers binaires (OTB_BINARY_DIR).

Il est récommandé d'installer et de compiler les fichiers binaires dans un répertoire autre que le répertoire des sources de l'OTB. Par exemple :

[7] http://www.cmake.org

```
mkdir OTB-bin
cd OTB-bin
ccmake ../OTB
```

Sous Windows, l'interface GUI de CMake est utilisée pour définir et construire les répertoires (Figure 2.1).

CMake se lance en mode intéractif, où il est possible de configurer certains paramètres et valider des options. L'étape de configuration permet alors de générer les fichiers de configuration.

2.1.2 Configurer l'OTB

La Figure 2.1 montre l'interface CMake pour UNIX et Windows.

Il est possible de choisir si l'on souhaite compiler les exemples (`OTB/Examples`) et les tests (`OTB/Testing`). Ceci est possible en positionnant les variables `BUILD_EXAMPLES=OFF` et `BUILD_TESTING=OFF`.

Ces exemples peuvent être utilisés pour se familiariser avec l'OTB. Les tests sont constitués d'un ensemble de petits programmes permettant de valider l'OTB. De plus, le composant `OTB-Applications` montrent des applications plus évoluées, utilisant notamment des IHM graphiques et de la visualisation.

2.2 Démarrer avec l'OTB

2.2.1 Hello World !

Cet exemple permet de montrer comment créer un petit programme, qui utilise la bibliothèque OTB (cet exemple se trouve dans le répertoire `OTB/Examples/Installation`). Le fichier `CMakeLists.txt` contient les lignes suivantes :

```
PROJECT(HelloWorld)

FIND_PACKAGE(OTB)
IF(OTB_FOUND)
  INCLUDE(${OTB_USE_FILE})
ELSE(OTB_FOUND)
  MESSAGE(FATAL_ERROR
          "OTB non trouvee. Positionner la variable OTB_DIR.")
ENDIF(OTB_FOUND)

ADD_EXECUTABLE(HelloWorld HelloWorld.cxx )

TARGET_LINK_LIBRARIES(HelloWorld OTBCommon)
```

La première ligne définie le nom du projet pour l'environnement Visual Studio (n'a aucun effet sous UNIX). La seconde ligne précise que la bibliothèque OTB est nécessaire pour compiler ce programme. Si elle n'est pas trouvée, un message d'erreur est émis. La ligne ADD_EXECUTABLE permet de définir le programme que l'on créée (le premier argument est le nom de l'exécutable, les suivants le(s) fichier(s) source(s) associé(s) à cet exécutable) La ligne TARGET_LINK_LIBRARIES précise quelles bibliothèques sont nécessaires pour générer cet exécutable.

Le code source se trouve dans le fichier
Examples/Installation/HelloWorld.cxx.

Cet exemple permet d'afficher sur la sortie standard un petit message de 'Bonjour'

```
#include "otbBonjour.h"
#include <iostream>
#include <iostream>
#include <string>

int main()
{
otb::Bonjour lBonjour;
std::string lString = lBonjour.getMessage();
std::cout << "Message : "<<lString<<std::endl;
return 0;
}
```

Ce code permet d'afficher sur la sortie standard, le message enregistre dans la classe otb::Bonjour.

Vous avez maintenant installé et compilé avec succès la bibliothèque OTB, et vous avez créé un programme simple se *linkant* avec cette bibliothèque. Si vous avez des difficultés, vous pouvez prendre contact avec les auteurs de ce document.

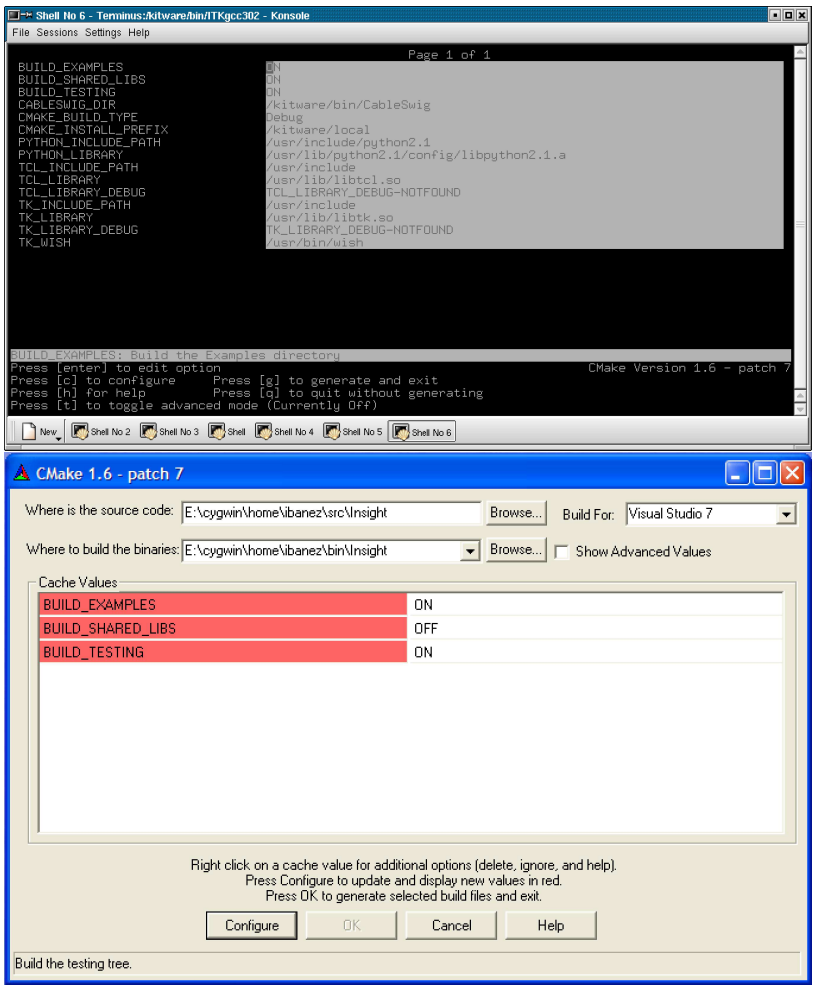


Figure 2.1: Interface CMake. En Haut) `ccmake`, interface sous UNIX. En Bas) `CMakeSetup`, la version MS-Windows pour les MFC.

Part II

Guide pour l'utilisateur

CHAPTER

THREE

Exemple de filtrage

Ce chapitre présente des exemples de filtrages .....

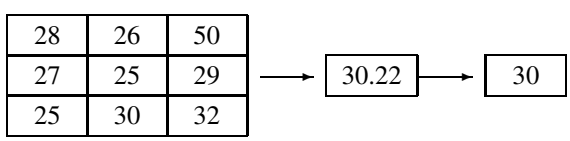
3.1 Filtrage de voisinage

BLA BLA....

3.1.1 Filtre Moyen

Le code source se trouve dans le fichier
`Examples/StartExamples/MeanImageFilter.cxx`.

BLA BLA...



Suite BLA BLA...

The header file corresponding to this filter should be included first.

```
#include "itkMeanImageFilter.h"
```

Then the pixel types for input and output image must be defined and, with them, the image types can be instantiated.

```
  typedef   unsigned char  InputPixelType;
```

```
typedef   unsigned char  OutputPixelType;

typedef itk::Image< InputPixelType,  2 >   InputImageType;
typedef itk::Image< OutputPixelType, 2 >   OutputImageType;
```

Using the image types it is now possible to instantiate the filter type and create the filter object.

```
typedef itk::MeanImageFilter<
             InputImageType, OutputImageType >  FilterType;

FilterType::Pointer filter = FilterType::New();
```

The size of the neighborhood is defined along every dimension by passing a `SizeType` object with the corresponding values. The value on each dimension is used as the semi-size of a rectangular box. For example, in $2D$ a size of $1,2$ will result in a 3×5 neighborhood.

```
InputImageType::SizeType indexRadius;

indexRadius[0] = atoi(argv[3]); // radius along x
indexRadius[1] = atoi(argv[4]); // radius along y

filter->SetRadius( indexRadius );
```

The input to the filter can be taken from any other filter, for example a reader. The output can be passed down the pipeline to other filters, for example, a writer. An update call on any downstream filter will trigger the execution of the mean filter.

```
filter->SetInput( reader->GetOutput() );
writer->SetInput( filter->GetOutput() );
writer->Update();
```

Figure 3.1 illustrates the effect of this filter on an image of a point avec voisinage de $10,10$ correspondant a un filtre de taille 21×21.

BLA BLA Mean filter

The typical effect of median filtration on a noisy digital image is a dramatic reduction in impulse noise spikes. The filter also tends to preserve brightness differences across signal steps, resulting in reduced blurring of regional boundaries. The filter also tends to preserve the positions of boundaries in an image.

Figure **??** below shows the effect of running the median filter with a 3x3 classical window size 1, 10 and 50 times. There is a tradeoff in noise reduction and the sharpness of the image when the window size is increased.

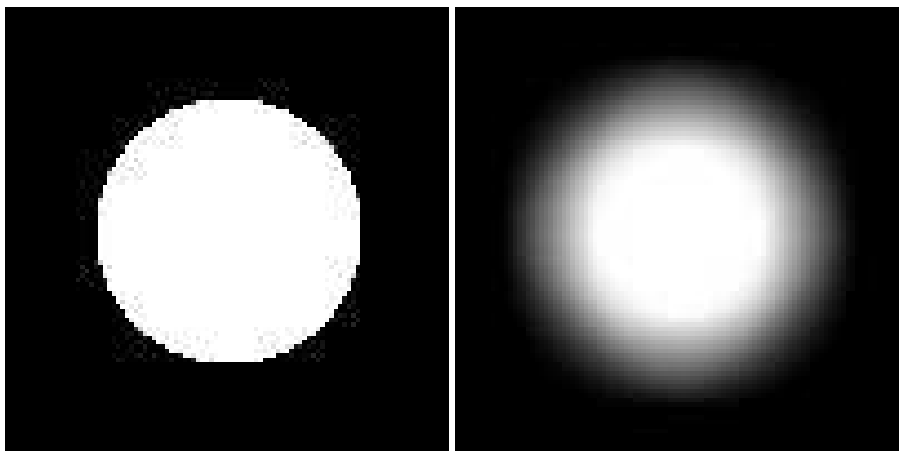


Figure 3.1: Effect of the MeanImageFilter on point.

Figure 3.2: Image de tests : temporaire !!!!

Part III

Guide pour le développeur