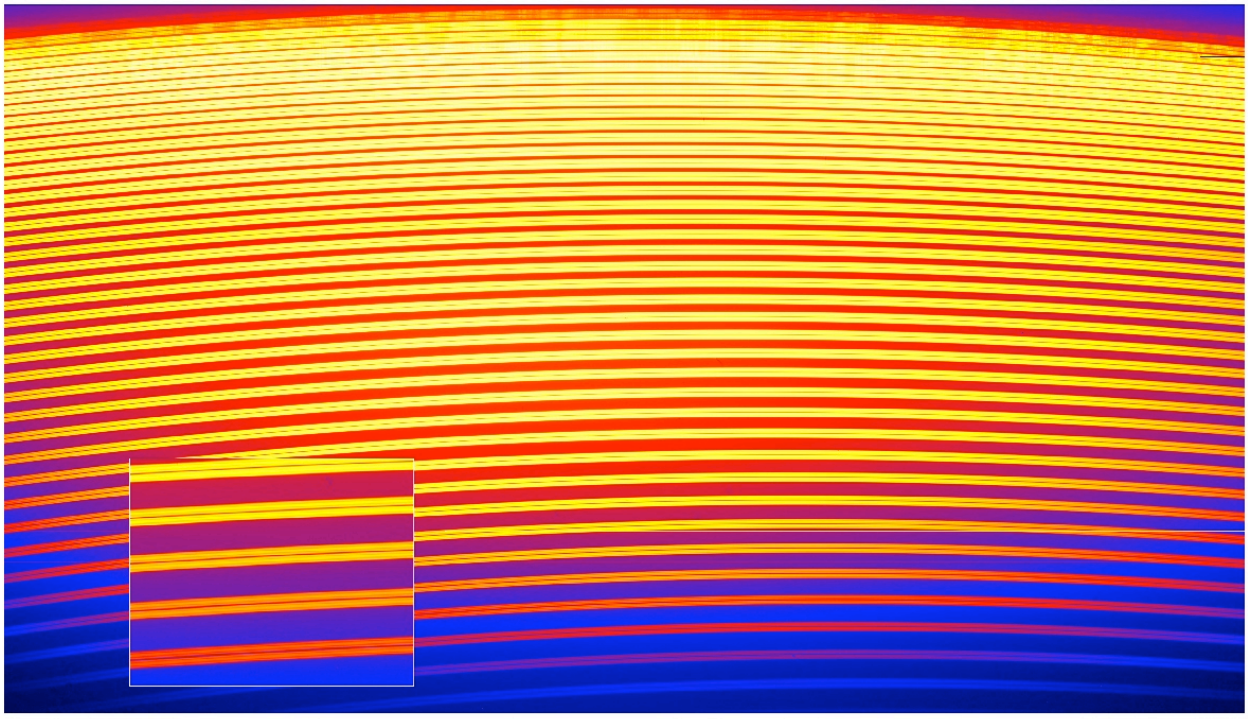
Canada-France-Hawaii Telescope Corporation
65-1238 Mamalahoa Hwy, Kamuela, Hawaii 96743 USA

Société du Télescope Canada-France-Hawaii
Telephone (808) 885-7944 Fax (808) 885 -7288

OPERA PIPELINE PROJECT

Getting Started



DOUG TEEPLE
Canada France Hawaii Telescope
Waimea, HI April 2013

Contents

1. Introduction

This document is intended to instruct new users in the operation of the OPERA (Open-source Pipeline for Echelle Reduction and Analysis) pipeline. OPERA is an open-source software reduction pipeline for echelle spectro-polarimetric data. OPERA is written in C and C++ and available in source form, ready for installation on a MacOSX or Linux platform.

This document details the operational steps involved in reducing espadons data with OPERA. It will show new users the commands that are available and the order in which these commands should be given. Next the document describes certain useful customization steps that may of value to remote installations. Appendix A gives brief download and installation instructions. Installation is relegated to an appendix since it is usually just done once, although it is the first step that needs to be taken. Appendix B shows the default directory structure of OPERA and the products and byproducts that OPERA produces, “Default” is the key here, because the actual product and byproduct structure is configurable.

2. OPERA Operational Steps

There are two major steps involved in obtaining spectrographic and polarimetric reduced data. First is calibration and second is reduction. Calibration consists of many smaller steps: creating master calibration images, calculation of gain and bias, geometry calibration, instrument profile calibration and wavelength calibration. Reduction consists. at a high level, of optimal intensity extraction and polarimetry. Calibration must precede reduction.

OPERA commands are designed to be dependency-driven and also to be hierarchical. Dependency-driven, in this context, means that if a calibration or reduction step requires a prior step in order to complete, that all prior commands will automatically be issued. Hierarchical means that you have on your demand, commands at a very high level, like *“Perform all calibrations for all modes and speeds of images”*. You may also issue commands at a finer granularity. For example *“Perform gain and bias calculation”* or *“Create reduction lists”* or *“Perform master calibrations”* or *“Perform Geometry Calibrations”*.

Basic commands are given now for illustrative purposes, the full command set is given in section 3.

2.1 Calibration

There are two major steps involved in performing a reduction. First is calibration and second is reduction. The highest level OPERA command to perform all calibration steps is:

```
opera DATADIR=<...> calibrations
```

This performs, in order:

1. reduction set creation
2. master calibration images (masterflats, mastercomparisons, masterfabryperots, masterbiases, normalizedflats)
3. gain and noise values
4. instrument profiles
5. geometry calibrations
6. wavelength calibrations

Each of these can be done individually, if desired.

2.2 Reduction

Reduction consists on a high level, of intensity optimal extraction and polarimetry. Other steps are signal to noise calculation, wavelength correction, normalization and (at CFHT) FITS product packaging. The basic command to perform all reductions is:

```
opera DATADIR=<...> reduce
```

This performs, in order:

1. extraction
2. Signal to noise calculation
3. normalization
4. telluric correction
5. FITS product creation

Polarimetry proceeds in parallel.

Again there are subcommands:

```
opera DATADIR=<...> intensity
```

```
opera DATADIR=<...> polarimetry
```

```
opera DATADIR=<...> spectrum FILE=...
```

The first commands perform as one would expect, while the third performs a reduction on a single image (for quicklook purposes while observing).

3. OPERA Commands

After installation, to find the command set, type:

`opera --help`

You will get something like this:

```
~/opera-1.0/] opera --help
 Usage: opera <options> <commands>
 Basic Commands:
     opera --help
     opera --version
 Options:
     opera -v | --verbose     -- verbose output
     opera -t | --trace       -- trace execution steps
     opera -d | --debug       -- debug output
     opera -p | --plot        -- generate plots
     opera -n | --noexecute   -- trace only, do not execute instructions (generates a script)
 Pipeline Commands:
     opera <specifier> reduce        -- parallel intensity and polarimetry reduction
     opera <specifier> intensity     -- parallel intensity reduction
     opera <specifier> polarimetry   -- parallel polarimetry reduction
     opera <specifier> spectrum FILE=<filename>     -- reduction of a single image
     opera <specifier> calibrate calibration calibrations   -- parallel calibrations
     opera <specifier> masters mastercalibrations serialcalibrations      -- serial calibrations
     opera <specifier> unlock        -- unlock a NIGHT directory
     opera <specifier> reductionset  -- create a reduction set
     opera <specifier> mastercalibrations    -- does all of the masters below
     opera <specifier> masterflats [--pick=0|1|2]
     opera <specifier> masterfabryperots [--pick=0|1|2]
     opera <specifier> gains
     opera <specifier> geometries
     opera <specifier> wavelengthcalibrations
     opera <specifier> profiles
     opera <specifier> clean         -- cleans all
     opera <specifier> cleantarget
     opera <specifier> cleanpol[arimetry]
     opera <specifier> cleanint[ensity]
     opera <specifier> cleancal[ibrations
     opera <specifier> clean[profs|gains|geoms]
```

```
    opera <specifier> cleanvisuals
    opera <specifier> cleanlogs
    opera <specifier> cleantmp
    opera <specifier> cleanit WHAT=<extension> i.e. geom gain prof eps s sn etc
Where <specifier> is of the form: NIGHT=QRUNID-MMMDD e.g. 08AQ04-Mar16 or DATADIR=<directory>
-- full directory path containing the data to reduce.
```

A bit overwhelming.

The first set of options are available with all commands. They are:

`--verbose`

Which means that modules are directed to print lots of information about the processing they are doing. The content of verbosity depends on the particular module.

`--trace`

This means to show the actual call to each module as it is being executed. Trace is handy when you are starting to debug modules as you can copy and paste the trace to the command line to debug a particular module. All the configuration and module execution parameters are instantiated.

`--debug`

This is used to print out large amounts of debug information from each module. Useful in early stages of development.

`--plot`

OPERA modules are capable of generating plots (either PNG or GNUPLOT eps) files in the “visuals” directory. These are very useful when porting to new instruments or new environments. Module writers are encouraged to generate profuse plots.

`--noexecute`

This command instructs OPERA to show the commands that it would execute, in the order that it would execute them, with all configuration and module parameters instantiated. This is handy for those who like the comfort of script-like execution, but the script is not suitable for execution in a production environment.

The basic calibration and reduction commands were given in Section 2. A few extra commands need explanation:

`opera DATADIR=<...> unlock`

When OPERA processing is started, the reduction or calibration is locked so that two individuals don't start processing the same data. If the process is aborted, the directories will remain locked. Use the unlock command to enable processing again.

`opera DATADIR=<...> clean...`

Disk space is limited. When final products have been created and distributed, the intermediate products may be cleaned off disk. There are obvious subcommands to clean specific items.

4. OPERA Customization

Please note that these customization steps cover only customization for the espadons instrument. Porting OPERA to a new instrument is beyond the scope of this document.

The main entry point to customization lies the the root Makefile in the OPERA harness:

`harness/Makefile`

This makefile contains the key variable “instrument”:

```
#################################################################
# IMPORTANT to define the instrument here, selects the set of
Makefiles to use...
#################################################################

instrument      := espadons
```

This selects the set of Makefiles to use in the harness that contain execution rules for a particular instrument. There are two further kingpins to OPERA customization. These are the “parameters” to modules, and the OPERA “configuration”. The parameters are stored in:

`harness/Makefile.parameters.espadons`

Parameters are intended to be module-specific processing parameters. In the design of OPERA we wanted to separate processing parameters from being hard-coded inside the module source code. The harness reads these parameters particular to each module and passes them the the module when it executes.

The parameters are different for each module and identified as in the sample below:

```
################################################################
# wavelength / telluric correction parameters
################################################################

wcal_threshold          := 0.5
wcal_sigma              := 2.0
FourierFilterWidth      := 4.0
thorium_argon_atlas     := thar_MM201006.dat
# -1 means all orders
wcal_ordertoplot        := 52

################################################################
# instrument profile
################################################################
ipDimensions            := 20 1 1 1
```

Second is the configuration. It contains location-dependent global definitions such as file paths. For example:

```
################################################################
#
# CFHT-specific paths
#
################################################################
ifeq ($(observatory),CFHT)

ifeq ($(OSTYPE),darwin)
      queuedir      := /data/espadons/
      outdir        := $(homedir)/opera/
else
      queuedir      := /data/niele/espadons/
      outdir        := /data/uhane5/opera/
endif

DATADIR        := $(queuedir)/$(NIGHT)/
ROOTDIR        := $(homedir)
NIGHT          := $$(basename $(DATADIR))
specdir        := $(outdir)/spectra/$$(basename $(DATADIR))/
reductiondir   := $(outdir)/reductions/$$(basename $(DATADIR))/
configdir      := $(operahomedir)/config/
calibrationdir := $(outdir)/calibrations/$$(basename $(DATA-
DIR))/
byproductsdir  := $(outdir)/byproducts/$$(basename $(DATADIR))/
processeddir   := $(outdir)/processed/$$(basename $(DATADIR))/
```

```
approveddir    := $(outdir)/approved/$$(basename $(DATADIR))/
tmpdir         := /tmp/$$(basename $(DATADIR))/
logdir         := $(outdir)/logs/$$(basename $(DATADIR))/
visualsdir     := $(outdir)/visuals/$$(basename $(DATADIR))/
endif
```

These definitions are important when moving to a new site and should be specified carefully before attempting to execute OPERA. Note the extensive use of conditional definitions and the use of Make variables.

An important feature of OPERA is scalability to production level capacity. Scalability by default means execute a single thread on the machine where OPERA is launched. However OPERA scales to execute multiple processes on multiple pieces of hardware simultaneously for production-level performance. Scalability is controlled as in the configurations shown below:

```
#########################################################################
#
# CFHT-specific reduction machine definitions. Modify these entries to
implement
# various machine use configurations.
#
#########################################################################

# this runs just on the local machine, launching 1 process.
MACHINES              := $${HOSTNAME}
LOADAVERAGES          := 1

# this runs just on the local machine, launching 4 simultaneous proc-
esses.
#MACHINES             := $${HOSTNAME}
#LOADAVERAGES         := 4

# this runs on a machine called "remoteserver", launching 12 simulta-
neous processes
# use this to launch from your desktop and do the heavy work on a
server.
#MACHINES             := remoteserver
#LOADAVERAGES         := 12

# this shows how to run on multiple machines, give a name and loadav-
erage for each,
```

```
# here we use 3 machines, polena having 8 cpus gets a heavier load...
# MACHINES            :=  hukilau naio polena
# LOADAVERAGES        :=  4      4          8
```

The user may uncomment the sample definitions (using site machine names) above to control execution in the site environment.

6. Conclusion

This document detailed the basic commands to install and execute OPERA on a MacOSX or Linux-based operating system. It described the basic calibration and reduction steps and the order in which calibration and reduction should proceed. Appendices gave download, installation and default directory structure.

7. Appendix A - Download / Installation

7.1 Downloading OPERA From SourceForge

OPERA source code and documentation are available on SourceForge at this location:

http://sourceforge.net/projects/opera-pipeline/?source=directory

On this page you will see a file called opera-1.0-<date>,zip, where <date> is the build date. Click on the latest build's link to retrieve the zip file containing the latest sources. Do not unzip it yet. If you select "Files" in the menu you can navigate to the opera-1.0 directory and also get documentation and an installer script called **INSTALL_OPERA-1.0** or **INSTALL_OPERA-1.0.sh**. You should download this file too. For Mac users there is also an xcode project called opera-1.0-xcodeproj.zip for convenience. If you want to use the xcode project, download and unzip it into your home directory and create a folder there called "build". You can get xcode from the Apple app store in iTunes for free.

Please download to your home directory.

We have tried to keep the dependencies to a minimum. That said, it doesn't make sense to write Fast Fourier Transforms or PNG graphics libraries by hand.

So, **before starting installation of opera**, be sure you have these installed: Please **DO NOT use macports, darwinports**, or fink as they put stuff in odd places and will cause you much pain later on. The libraries below are small and build and install from source very easily.

---------- gotta have these ----------

1. cfitsio -- (mandatory) read and write FITS images

    -- http://heasarc.gsfc.nasa.gov/docs/software/fitsio/fitsio.html

    Download the Linux version into your home directory and do the simplified steps:

    cd cfitsio

    ./configure

    make

    sudo make install

2. FFTW3 -- (mandatory)FFT Library

    -- http://www.fftw.org/

        Download the Linux version into your home directory and do the simplified steps:

    cd fftw-3.3

    ./configure

    make

    sudo make install

3. libPNG -- (mandatory) used by the tool operaFITStoPNG and some visualization tools

-- http://www.libpng.org

4. XCode -- FOR MAC USERS ONLY - used to browse the opera-1.0 project on a mac and also

to get a gcc compiler to build the above libraries...

- http://developer.apple.com

- you must register as a mac developer

- download Code and install it

- download the command line tools extras and install it (includes gcc or llvm-gcc)

---------- you can stop here ----------

5. fitsverify -- (optional) used to check the validity of FITS output files

-- http://heasarc.gsfc.nasa.gov/docs/software/ftools/fitsverify/

6. gnuplot 4.4 -- (optional) used by some modules, but a test is made to see if gnuplot is installed,

so operation will proceed without it (and without the plots)

7. libfreetype -- (optional) used by the visualization tools to label graphs

-- http://www.freetype.org/

---------- no need to go farther ----------

8. doxygen -- (optional) used if you need to regenerate documentation

-- http://www.doxygen.org

9. graphviz -- (optional) used by doxygen if the doc/Doxyfile has HAVE_DOT = YES,

meaning that you want to generate callgraphs in the documentation.

-- http://www.graphviz.org

10. SOFA -- (optional) This is the IAU Standards of Fundamental Astronomy (SOFA) Libraries product

-- http://www.iausofa.org/

11. JPL -- (optional) Solar System Ephemeris

-- http://www.cv.nrao.edu/~rfisher/Ephemerides/ephem_descr.html

The complete list is found in the DEPENDENCIES readme in the opera home directory. Please compile and install these libraries from source so that the include and lib directories go to /usr/local/include and /usr/local/lib/.

There is a pre-built xcode project available also for those who want a quick start on the Mac. Download the opera-1.0-xcodeproj.zip zip file and unzip it in your home directory. Install the opera-1.0 folder in your home directory, and then open the project with xcode and you should see all the opera files in the project window. This is a Mac-only project, DO NOT try to use it on Linux..

7.2 Installing OPERA

First, change directory to your home directory. Get the latest build from Sourceforge.

On a Mac, the opera-1.0.zip unzips itself in the Downloads folder. The easiest thing to do is rezip it (CTRL-Select -> Compress) or jut move it to your $HOME directory.

If you already have a copy of opera-1.0:

```
rm -rf opera-1.0/
```

Then type:

```
. ./INSTALL_OPERA-1.0 ./opera-1.0.zip
```

or

```
. ./INSTALL_OPERA-1.0.sh ./opera-1.0.zip
```

Please do this carefully, it is:

dot space dot/INSTALL_OPERA-1.0 *space dot*/opera-1.0.zip

Next type either:

```
. ./setup.sh
```

or:

```
export opera=$HOME/opera-1.0
```

```
cd $opera
```

For convenience you can put the export in your .bashrc or profile.

8. Appendix B - Where's My Stuff?

This appendix documents *default* locations of the OPERA source tree and of the locations of OPERA execution products and byproducts.. "Default" is key here, as OPERA is highly customizable.

8.1 The Stuff that OPERA Creates

Key locations are in the configuration Makefile (Makefile.configuation.espadons) are:

`DATADIR` - this is a directory containing the set of all calibration and object images to be considered as a complete reduction. By default on a Mac the base of this should be: /data/espadons/

`NIGHT` - this is the tail of the `DATADIR`, which gives a unique identifier for the set of data.

The output by default goes to: $HOME/opera/

`specdir` - this is wherr the spectra are placed $HOME/opera/spectra/

`visualsdir` - this is where the .eps plots are placed $HOME/opera/visual/

`configdir` - this is where configuration files for an instrument should be placed $HOME/opera-1.0/config/

`calibrationdir` - this is where the output calibration files are placed $HOME/opera/calibrations/

byproductsdir - this is where byproducts (intermediate files) are place. By-products are not distributed to PIs in the CFHT context. $HOME/opera/byproducts/

`processdir` - this is a CFHT-specific directory which stores processed data on a program basis $HOME/opera/processed/

`approveddir` - this is a CFHT-specific directory which stores processed data on a program basis after approval $HOME/opera/approved/

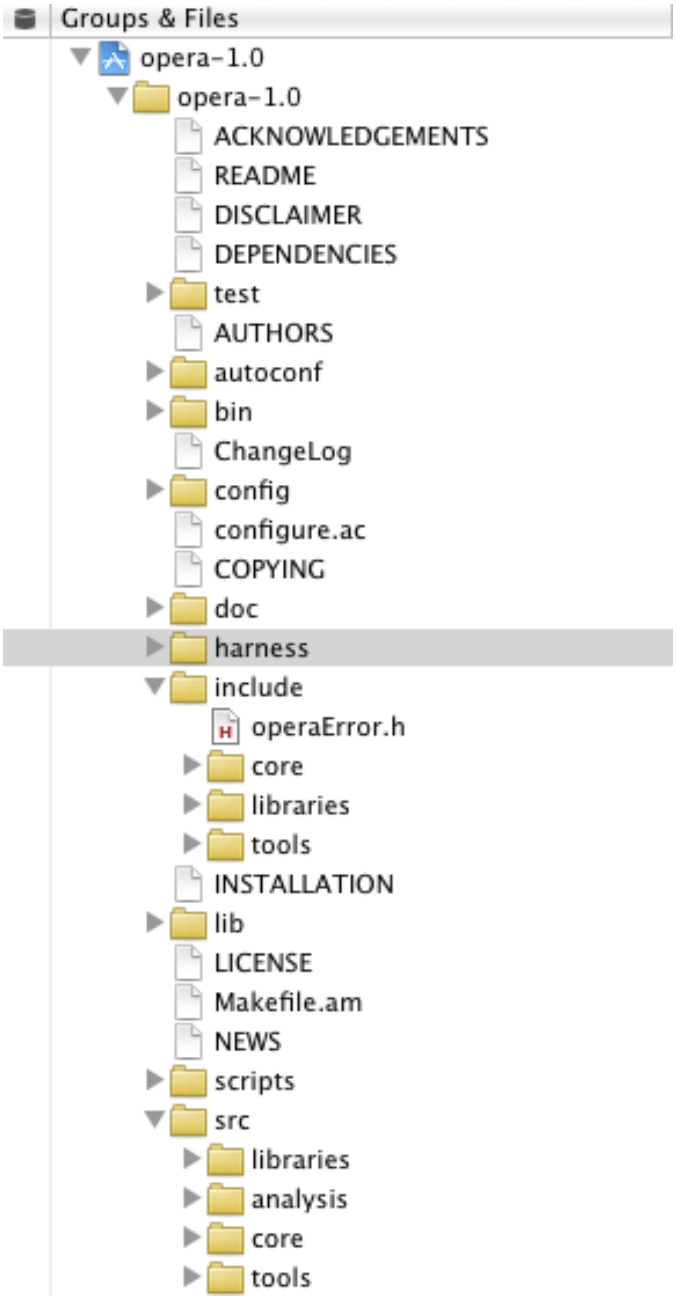
`tmpdir` - this is where temporary files amy be stored by a module /tmp/

`logdir` - all execution steps and error logs are stored here $HOME/opera/logs/

`analysesdir` - all anaysis output is stored here $HOME/opera/analyses/

8.2 OPERA Directory Structure - The Stuff of OPERA

The overall OPERA directory structure (taken from the XCode project window) is shown here:



Without going in to too much detail, the overall structure is:

1. `test` - contains test cases for libraries
2. `bin` - where binaries are stored
3. `config` - contains static espadons configuration (calibration) data
4. `doc` - contains the OPERA documentation including doxygen
5. `harness` - contains the Makefiles comprising the harness
6. `include` - contains include files for the modules (core), libraries and tools
7. `lib` - contains library binaries
8. `scripts` - contains handy bash scripts
9. `src` - contains C/C++ source code for the modules (core), libraries and tools.