OSKAR Applications

Version history:

Revision	Date	Modification
1	2012-04-20	Creation.
2	2012-05-15	[2.0.1] Added description of binaries to set and display OSKAR settings parameters, and an example shell script.
3	2012-06-19	[2.0.2] Updated description of binaries to include those that can be used to query an OSKAR binary file, and export visibilities to a Measurement Set.
4	2012-07-27	[2.0.4] Updated description of binaries to include display of CUDA system information.
5	2013-02-26	[2.2.0] New applications: oskar_fits_image_to_sky_model, oskar_image_stats, oskar_visibilties_add. Moved description of MATLAB interface into its own document.
6	2013-11-16	[2.3.0] Renamed applications handling OSKAR visibility binary files from using the word *visibilities* to *vis*. New applications: oskar_vis_add_noise, oskar_vis_summary.
7	2014-02-26	[2.4.0] Fixed a settings key in the example BASH script.
8	2014-07-16	[2.5.0] New application: oskar_fit_element_data.
9	2014-09-09	[2.5.1] Updated description for oskar_vis_to_ms application, which now supports concatenation of visibility files.

1 Introduction

This document briefly summarises the current applications included in the OSKAR package. It assumes that OSKAR has already been built and installed.

2 Application Binaries

Currently, there are 16 OSKAR application binaries available, listed below in alphabetical order. Applications that can be used to perform simulations with OSKAR are marked in **bold**.

1. **`oskar`**
2. `oskar_binary_file_query`
3. `oskar_cuda_system_info`
4. `oskar_fit_element_data`
5. `oskar_fits_image_to_sky_model`
6. `oskar_image_stats`
7. `oskar_image_summary`
8. `oskar_imager`
9. `oskar_settings_get`
10. `oskar_settings_set`
11. **`oskar_sim_beam_pattern`**
12. **`oskar_sim_interferometer`**
13. `oskar_vis_add`
14. `oskar_vis_add_noise`
15. `oskar_vis_summary`
16. `oskar_vis_to_ms`

For a description of each of these applications, please refer to their respective numbered subsections. When running applications, usage syntax as well as some usage examples can be obtained by specifying the `--help` flag as a command line argument to the binary (e.g. `$ oskar --help`). The OSKAR package version number from which the binary was built can be obtained for all applications by specifying the `--version` flag.

Application binaries are built into the `<build directory>/apps` folder and installed into `/usr/local/bin` by default.

2.1 oskar

This application provides a simple graphical user interface that can be used to configure and run simulations. It can be started with the following syntax:

```
$ oskar [settings file path]
```

2.2 oskar_binary_file_query

This utility displays a summary of the contents of an OSKAR binary file, and can be run using the syntax:

```
$ oskar_binary_file_query <binary file path>
```

2.3 oskar_cuda_system_info

This utility displays a summary of the installed CUDA hardware. It takes no command line arguments.

2.4 oskar_fit_element_data

This application must be used if numerically-defined element pattern data should be used in a simulation. It performs spline fitting to tabulated data, and stores the fitted coefficients to files inside the telescope model. All options are configured using the element fit group of the specified settings file. The application is run using the following syntax:

```
$ oskar_fit_element_data <settings file path>
```

Note that this application can be configured and run via the `oskar` GUI application, described in section 2.1.

2.5 oskar_fits_image_to_sky_model

This utility can be used to convert a standard radio astronomy FITS image (made using the orthographic projection) to an OSKAR sky model file. It takes the following command line syntax:

```
$ oskar_fits_image_to_sky_model [OPTIONS] <Input FITS file>
                                          <Output sky model file>
```

`[OPTIONS]` consists of flags to specify how much of the input image is converted. The downsample factor, noise floor, and minimum peak fraction can all be set here.

2.6 oskar_image_stats

This utility will evaluate a number of statistics from one or more OSKAR binary image files. The minimum, maximum, mean, variance, standard deviation, and RMS pixel values for each image are output to the terminal. The application is run with the following syntax:

```
$ oskar_image_stats [OPTIONS] <image file(s)>
```

`[OPTIONS]` consists of flags for specifying the time, channel and polarisation index of the image plane from which to obtain statistics.

2.7 oskar_image_summary

This utility displays a summary of the contents of an OSKAR binary image in the terminal. The application is run with the following syntax:

```
$ oskar_image_summary <image file>
```

2.8 oskar_imager

An application that can be used to make raw (dirty) images, in FITS or OSKAR binary image format, from visibility data stored in OSKAR binary visibility data files. Images or image cubes are made using a GPU-based DFT algorithm. All options are configured in the image group of a specified OSKAR settings file, and the imager is run with the following syntax:

```
$ oskar_imager <settings file path>
```

Note that this application can be configured and run via the `oskar` GUI application, described in section 2.1.

2.9 oskar_settings_get

Intended for use when scripting OSKAR, this utility reads the value of a specified settings key in an OSKAR settings file. The settings value is written to the terminal standard output. This application is run using the following syntax:

```
$ oskar_settings_get <settings file path> <key>
```

2.10 oskar_settings_set

Intended for use when scripting OSKAR, this utility can be used to write settings to an OSKAR settings file. The value of the setting, the settings key and settings file are specified on the command line according to the following syntax:

```
$ oskar_settings_set <settings file path> <key> <value>
```

An example shell script that makes use of `oskar_settings_set` is shown later in this document. Please see the accompanying documentation for details of the options that can be set in OSKAR-2 settings files.

2.11 oskar_sim_beam_pattern

This is a command line application for simulating station beam patterns, which is configured by providing an OSKAR settings file as the first command line argument. Beam patterns produced by this application are simulated using the same algorithms used in the interferometry simulation. The application is run with the following syntax:

```
$ oskar_sim_beam_pattern <settings file path>
```

Note that this application can be configured and run via the `oskar` GUI application, described in section 2.1.

2.12 oskar_sim_interferometer

This is a command line application for simulating interferometer data. Visibility data sets produced by the simulator are written in Measurement Set or OSKAR binary visibility format. The simulation is configured using a variety of options, which are specified in an OSKAR settings file provided as the first command line argument:

```
$ oskar_sim_interferometer <settings file path>
```

Note that this application can be configured and run via the `oskar` GUI application, described in section 2.1.

2.13 oskar_vis_add

This application combines two or more OSKAR binary visibility files. It is intended for combining simulations made with different sky model components, so the visibility data files being combined must have been generated using identical telescope configurations and observation parameters (i.e.

share common baseline coordinates, time and frequency axes). The application is run with the following syntax:

```
$ oskar_vis_add [OPTIONS] <OSKAR visibility files...>
```

`[OPTIONS]` consists of flags for specifying the output (combined) visibility data file name, and a flag for supressing log messages.

2.14 oskar_vis_add_noise

This application adds noise to the specified OSKAR binary visibility file(s). The noise to be added is configured according to the noise settings found in the interferometer group of the provided OSKAR simulation settings file (for details of these settings please refer to the OSKAR-Settings documentation). The application is run with the following syntax:

```
$ oskar_vis_add_noise [OPTIONS] <OSKAR visibility files...>
```

`[OPTIONS]` consists of flags for specifying the settings file in which the noise parameters are defined, whether noise should be added in-place or to a copy of the input visibility file(s), and a flag to enable verbose output.

2.15 oskar_vis_summary

This application prints a summary of the data contained within an OSKAR visibility binary file. The application is run with the following syntax:

```
$ oskar_vis_summary [OPTIONS] <OSKAR visibility files...>
```

`[OPTIONS]` consists of flags to display the settings used to generate the visibility file and the run log generated during the simulation.

2.16 oskar_vis_to_ms

This application can be used to convert one or more OSKAR visibility binary file(s) to Measurement Set format. If more than one input OSKAR visibility file is provided, they are concatenated. The application is run with the following syntax:

```
$ oskar_vis_to_ms [OPTIONS] <OSKAR visibility files...>
```

3 Example BASH Shell Script

This section shows an example shell script (written for the BASH shell on Linux) that was used to run the simulations for the examples described in the '*OSKAR Theory of Operation*' document. It procedurally generates a sky model (containing a single source) at different locations and in different polarisation states, runs the simulation, and generates images in Stokes Q and U. The simulation results are discussed in that document: this section only exists to show an example of the scripting capability of OSKAR.

```
#!/bin/bash

# Name of temporary INI file and sky file.
INI=temp.ini
SKY=temp.sky

# Name of binary.
OSKAR_SET=oskar_settings_set

# General settings.
$OSKAR_SET $INI sky/oskar_sky_model/file $SKY
$OSKAR_SET $INI observation/start_frequency_hz 100000000
$OSKAR_SET $INI observation/start_time_utc "21-09-2000 00:00:00.000"
$OSKAR_SET $INI observation/length 12:00:00.000
$OSKAR_SET $INI observation/num_time_steps 24
$OSKAR_SET $INI telescope/input_directory \
        ../../data/telescope/hexagonal_regular_small_25x2587
$OSKAR_SET $INI telescope/longitude_deg 0
$OSKAR_SET $INI telescope/aperture_array/array_pattern/enable false
$OSKAR_SET $INI telescope/aperture_array/element_pattern/enable_numerical false
$OSKAR_SET $INI telescope/aperture_array/element_pattern/functional_type "Geometric dipole"
$OSKAR_SET $INI interferometer/image_output true
$OSKAR_SET $INI image/fits_image true

# Define source Stokes Q and U values to use.
STOKES_VAL=("1 0" "-1 0" "0 1" "0 -1")

# Define signs of source Stokes parameters.
STOKES_SIGN=("+" "-" "+" "-")

# Define Stokes image types to make for each source.
STOKES_TYPE=("Q" "Q" "U" "U")

# Define source coordinates (0, 87) and (90, 87).
COORD_RA=("0" "90")
COORD_DEC=("87" "87")

# Set telescope at the North Pole.
$OSKAR_SET $INI telescope/latitude_deg 89.9

# Loop over source positions.
for (( COORD_INDEX=0; COORD_INDEX<${#COORD_RA[@]}; COORD_INDEX++ )); do
    # Get RA and Dec values.
    RA=${COORD_RA[$COORD_INDEX]}
    DEC=${COORD_DEC[$COORD_INDEX]}

    # Set RA and Dec of phase centre.
    $OSKAR_SET $INI observation/phase_centre_ra_deg $RA
    $OSKAR_SET $INI observation/phase_centre_dec_deg $DEC

    # Loop over source Stokes parameters.
    for (( POL_INDEX=0; POL_INDEX<${#STOKES_VAL[@]}; POL_INDEX++ )); do
```

```
        # Create the sky model.
        echo "$RA $DEC 1 ${STOKES_VAL[$POL_INDEX]}" > $SKY

        # Set the image type.
        $OSKAR_SET $INI image/image_type ${STOKES_TYPE[$POL_INDEX]}

        # Set the image root filename.
        $OSKAR_SET $INI image/root_path \
                "RA_${RA}_Stokes${STOKES_SIGN[$POL_INDEX]}"

        # Run the interferometer simulation.
        oskar_sim_interferometer $INI
    done
done

# Define source coordinates (1, 0)
COORD_RA=("1")
COORD_DEC=("0")

# Set telescope at the Equator.
$OSKAR_SET $INI telescope/latitude_deg 0

# Loop over source positions.
for (( COORD_INDEX=0; COORD_INDEX<${#COORD_RA[@]}; COORD_INDEX++ )); do
    # Get RA and Dec values.
    RA=${COORD_RA[$COORD_INDEX]}
    DEC=${COORD_DEC[$COORD_INDEX]}

    # Set RA and Dec of phase centre.
    $OSKAR_SET $INI observation/phase_centre_ra_deg $RA
    $OSKAR_SET $INI observation/phase_centre_dec_deg $DEC

    # Loop over source Stokes parameters.
    for (( POL_INDEX=0; POL_INDEX<${#STOKES_VAL[@]}; POL_INDEX++ )); do
        # Create the sky model.
        echo "$RA $DEC 1 ${STOKES_VAL[$POL_INDEX]}" > $SKY

        # Set the image type.
        $OSKAR_SET $INI image/image_type ${STOKES_TYPE[$POL_INDEX]}

        # Set the image root filename.
        $OSKAR_SET $INI image/root_path \
                "Equator_RA_${RA}_Stokes${STOKES_SIGN[$POL_INDEX]}"

        # Run the interferometer simulation.
        oskar_sim_interferometer $INI
    done
done

# Clear temporary files.
rm -f $INI $SKY

# View results.
ds9 RA_0*.fits &
ds9 RA_90*.fits &
ds9 Equator_RA_1*.fits &
```