
MIPP and SMHI/DMI Common Processing Environment

Release v1.0.0

Lars Orum Rasmussen

May 26, 2017

Contents

1	python-mipp an introduction	3
2	Calibration comments	7
3	The mipp API	9
4	Indices and tables	13
	Bibliography	15
	Python Module Index	17

This is a presentation of:

python-mipp an introduction

mipp is a Meteorological Ingest-Processing Package (<http://github.com/loerum/mipp>).

It's a Python library and it's main task is to convert low level satellite data into a format understood by mpop (<http://github.com/mraspaud/mpop>). The primary purpose is to support Geostationary satellite data (level 1.5) but there is also support for the reading of some polar orbiting SAR data (see below).

A more sophisticated interface to satellite data objects is supported by mpop.

Currently it handles data from all current Meteosat Second Generation (MSG) satellites, Meteosat 7, GOES 11-15, MTSAT's, and GOMS, all as retrieved via EUMETCast:

```
L-000-MTP___-MET7_____ -00_7_057E-PRO_____ -201002261600-__
L-000-MTP___-MET7_____ -00_7_057E-000001_____ -201002261600-C_
L-000-MTP___-MET7_____ -00_7_057E-000002_____ -201002261600-C_
L-000-MTP___-MET7_____ -00_7_057E-000003_____ -201002261600-C_
...
...
L-000-MSG2___-GOES11_____ -00_7_135W-PRO_____ -201002261600-__
L-000-MSG2___-GOES11_____ -00_7_135W-000001_____ -201002261600-C_
L-000-MSG2___-GOES11_____ -00_7_135W-000002_____ -201002261600-C_
L-000-MSG2___-GOES11_____ -00_7_135W-000003_____ -201002261600-C_
...
...
```

In addition mipp handles Synthetic Aperture Radar (SAR) data from Terrscan-X, Cosmo-Sky Med, and Radarsat 2.

mipp will:

- Decompress XRIT files (if Eumetsat's xRITDecompress is available). Software to uncompress HRIT/XRIT can be obtained from EUMETSAT (register and download the [Public Wavelet Transform Decompression Library Software](#)). Please be sure to set the environment variable XRIT_DECOMPRESS_PATH to point to the full path to the decompression software, e.g. /usr/bin/xRITDecompress. Also you can specify where the decompressed files should be stored after decompression, using the environment variable XRIT_DECOMPRESS_OUTDIR. If this variable is not set the decompressed files will be found in the same directory as the compressed ones.
- Decode/strip-off (according to [\[CGMS\]](#), [\[MTP\]](#), [\[SGS\]](#)) XRIT headers and collect meta-data.
- Catenate image data into a numpy-array.
 - if needed, convert 10 bit data to 16 bit

- if a region is defined (by a slice or center, size), only read what is specified.

Note:

- MET7: not calibrated.
 - GOES, METSAT: calibration constants to Kelvin or Radiance (not Reflectance).
-

Code Layout

xrit.py

It knows about the generic HRIT/XRIT format

- `headers = read_headers(file_handle)`

MTP.py

It knows about the specific format OpenMTP for MET7

- `mda = read_metadata(prologue, image_file)`

SGS.py

It knows about the specific format Support Ground Segments for GOES and MTSAT

- `mda = read_metadata(prologue, image_files)`

sat.py

It knows about satellites base on configurations files. It returns a slice-able object (see below).

- `image = load('met7', time_stamp, channel, mask=False, calibrated=True)`
- `image = load_files(prologue, image_files, **kwarg)`

slicer.py

It knows how to slice satellite images (return from `load(...)`). It returns meta-data and a numpy array.

- `mda, image_data = image[1300:1800, 220:520]`
- `mda, image_data = image(center, size)`

Utilities

cfg.py

It knows how to read configuration files, describing satellites (see below).

convert.py

10 to 16 byte converter (uses a C extension)

bin_reader.py

It reads binary data (network byte order)

- `read_uint1(buf)`
- `read_uint2(buf)`
- `read_float4(buf)`
- ...

mda.py

A simple (anonymous) metadata reader and writer

geosnav.py

It will convert from/to pixel coordinates to/from geographical longitude, latitude coordinates.

Example definition of a satellite

```
# An item like:
#   name = value
# is read in python like:
#   try:
#       name = eval(value)
#   except:
#       name = str(value)
#

[satellite]
satname = 'meteosat'
number = '07'
instruments = ('mviri',)
projection = 'geos(57.0)'

[mviri-level2]
format = 'mipp'

[mviri-level1]
format = 'xrit/MTP'
dir = '/data/eumetcast/in'
filename = 'L-000-MTP____-MET7_____-%(channel)s_057E-%(segment)s-%Y%m%d%H%M-__'

[mviri-1]
name = '00_7'
frequency = (0.5, 0.7, 0.9)
resolution = 2248.49
size = (5000, 5000)

[mviri-2]
name = '06_4'
frequency = (5.7, 6.4, 7.1)
resolution = 4496.98
size = (2500, 2500)

[mviri-3]
name = '11_5'
frequency = (10.5, 11.5, 12.5)
resolution = 4496.98
size = (2500, 2500)
```

Usage

```
import xrit

image = xrit.sat.load('meteosat07', datetime(2010, 2, 1, 10, 0), '00_7', mask=True)
mda, image_data = image(center=(50., 10.), size=(600, 500))
print mda
fname = './' + mda.product_name + '.dat'
print >>sys.stderr, 'Writing', fname
fp = open(fname, "wb")
image_data.tofile(fp)
fp.close()
```

Examples of the usage of some lower level tools

Here an example how to get the observation times (embedded in the ‘Image Segment Line Quality’ record) of each scanline in a segment:

```
import mipp.xrit.MSG

segfile = "/local_disk/data/MSG/HRIT/H-000-MSG3__-MSG3_____-WV_062____-000002____-
↪201311211300-__"
lineq = mipp.xrit.MSG.get_scanline_quality(segfile)
print lineq[0]

(465, datetime.datetime(2013, 11, 21, 13, 1, 48, 924000), 1, 1, 0)
```

A script, process_fsd

The script is intended for work on other geostationary data than the MSG (Meteosat) data, the so-called Foreign Satellite Data (FSD). That is e.g. GOES, MTSAT and COMS.

```
process_fsd --check-satellite <prologue-file>
    check if we handle this satellite

process_fsd --check [-l] <prologue-file>
    check if number of image segments are as planned
    -l, list corresponding image segment files

process_fsd --decompress [-o<output-dir>] <file> ... <file>
    decompress files to output-dir (default is working directory)
    -l, list decompressed files

process_fsd --metadata <prologue-file> <image-segment> ... <image-segment>
    print meta-data

process_fsd [-o<output-dir>] <prologue-file> <image-segment> ... <image-segment>
    it will binary dump image-data and ascii dump of meta-data)
```

MSG series

The calibration of the meteosat second generation satellites is done according to the Eumetsat documents [\[refl\]](#), [\[bt\]](#).

Reflectances

The visible and near infrared channels are calibrated according to the following formula:

$$r = R / I$$

where

- r is the bidirectional reflectance factor
- R is the measured radiance
- I is the solar irradiance

R is derived from the xRIT data, and I is given in [\[refl\]](#).

In [\[refl\]](#) the additional following corrections are applied:

- sun-earth distance correction
- cosine of the solar zenith angle.

MIPP

```
exception mipp.CalibrationError
exception mipp.ConfigReaderError
exception mipp.DecodeError
exception mipp.MippError
exception mipp.NavigationError
exception mipp.NoFiles
exception mipp.ReaderError
exception mipp.UnknownSatellite

mipp.strptime()
    string, format -> new datetime parsed from a string (like time.strptime()).
```

Metadata

```
class mipp.mda.Metadata

    dont_eval = ('satnumber',)
    ignore_attributes = ()
    read(file_name)
        Read until empty line, 'EOH' or 'EOF'.
    save(file_name)
    token = ':'

mipp.mda.mslice(mda)
```

Configuration

```
mipp.cfg.read_config(satname, instrument='')
```

Logging

```
class mipp.log.NullHandler
```

Empty handler.

```
    emit(record)
```

Record a message.

```
mipp.log.debug_on()
```

Turn debugging logging on.

```
mipp.log.get_logger(name)
```

Return logger with null handle

```
mipp.log.logging_off()
```

Turn logging off.

```
mipp.log.logging_on(level=None)
```

Turn logging on.

XRIT input layer

MSG

This module will read MSG level1.5 files, format documented in: ‘MSG Level 1.5 Image Data Format Description’, EUM/MSG/ICD/105, v5A, 22 August 2007

```
mipp.xrit.MSG.read_metadata(prologue, image_files, epilogue)
```

Selected items from the MSG prologue file.

GOMS

Read Electro L N1 HRIT files.

```
mipp.xrit.GOMS.read_epiheader(fp)
```

```
mipp.xrit.GOMS.read_metadata(prologue, image_files, epilogue)
```

Selected items from the Electro L N1 prolog file.

```
mipp.xrit.GOMS.read_proheader(fp)
```

MTP

This module will read satellit data files in OpenMTP format (eg. Meteosat-7 prolog file). Format described in: ‘The Meteosat Archive; Format Guide No. 1; Basic Imagery: OpenMTP Format’; EUM FG 1; Rev 2.1; April 2000

```
mipp.xrit.MTP.read_metadata(prologue, image_files)
```

Selected items from the Meteosat-7 prolog file.

SGS

This module will read satellit data files in SGS (Support Ground Segments) format (eg. GOES, MTSAT). Format described in: ‘MSG Ground Segment LRIT/HRIT Mission Specific Implementation’; EUM/MSG/SPE/057; Issue 6; 21 June 2006

`mipp.xrit.SGS.read_metadata` (*prologue, image_files*)

Selected items from the GOES image data files (not much information in prologue).

xrit

This module will read LRIT/HRIT headers. Format described in: “LRIT/HRIT Global Specification”; CGMS 03; Issue 2.6; 12 August 1999 “MSG Ground Segment LRIT/HRIT Mission Specific Implementation”; EUM/MSG/SPE/057; Issue 6; 21 June 2006

`mipp.xrit._xrit.read_prologue` (*file_name*)

`mipp.xrit._xrit.read_epilogue` (*file_name*)

`mipp.xrit._xrit.read_imagedata` (*file_name*)

`mipp.xrit._xrit.read_gts_message` (*file_name*)

`mipp.xrit._xrit.read_mpef` (*file_name*)

`mipp.xrit._xrit.read_mpef_clm` (*file_name*)

`mipp.xrit._xrit.decompress` (*infile, outdir='.'*)

Will decompress an XRIT data file and return the path to the decompressed file. It expect to find Eumetsat’s xRITDecompress through the environment variable XRIT_DECOMPRESS_PATH

`mipp.xrit._xrit.list` (*file_name, dump_data=False*)

bin_reader

`mipp.xrit.bin_reader.read_cds_expanded_time` (*buf*)

`mipp.xrit.bin_reader.read_cds_time` (*buf*)

`mipp.xrit.bin_reader.read_cuc_time` (*buf, coarse, fine*)

`mipp.xrit.bin_reader.read_float4` (*buf*)

`mipp.xrit.bin_reader.read_float8` (*buf*)

`mipp.xrit.bin_reader.read_int2` (*buf*)

`mipp.xrit.bin_reader.read_int4` (*buf*)

`mipp.xrit.bin_reader.read_uint1` (*buf*)

`mipp.xrit.bin_reader.read_uint2` (*buf*)

`mipp.xrit.bin_reader.read_uint4` (*buf*)

`mipp.xrit.bin_reader.read_uint8` (*buf*)

loader

`class mipp.xrit.loader.ImageLoader` (*mda, image_files, mask=False, calibrate=False*)

`raw_slicing` (*item*)

Raw slicing, no rotation of image.

convert

```
mipp.xrit.convert.dec10216 (in_buffer)  
mipp.xrit.convert.hrpt_dec10216 (in_buffer)
```

sat

```
mipp.xrit.sat.load_meteosat07 (time_stamp, channel, **kwarg)  
mipp.xrit.sat.load_meteosat09 (time_stamp, channel, **kwarg)  
mipp.xrit.sat.load_goes11 (time_stamp, channel, **kwarg)  
mipp.xrit.sat.load_goes12 (time_stamp, channel, **kwarg)  
mipp.xrit.sat.load_goes13 (time_stamp, channel, **kwarg)  
mipp.xrit.sat.load_mtsat1r (time_stamp, channel, **kwarg)  
mipp.xrit.sat.load_mtsat2 (time_stamp, channel, **kwarg)  
mipp.xrit.sat.load_electrol (time_stamp, channel, **kwarg)  
mipp.xrit.sat.load_himawari8 (time_stamp, channel, **kwarg)  
mipp.xrit.sat.load (satname, time_stamp, channel, **kwarg)  
mipp.xrit.sat.load_files (prologue, image_files, epilogue=None, **kwarg)
```

Metadata

```
class mipp.xrit.mda.Metadata  
  
    ignore_attributes = ('line_offset', 'first_pixel', 'coff', 'loff', 'image_data', 'boundaries')  
    token = ':'
```

XSAR input layer

Cosmo Sky-med

Radarsat-2

Terra-SAR X

sat

```
mipp.xsar.sat.load (satname, time_stamp, channel, **kwarg)
```

Metadata

```
class mipp.xsar.mda.Metadata  
  
    ignore_attributes = ('data', 'calibrate', 'tiepoints')  
    token = ':'
```

CHAPTER 4

Indices and tables

- `genindex`
- `modindex`
- `search`

Bibliography

- [CGMS] LRIT/HRIT Global Specification; CGMS 03; Issue 2.6; 12 August 1999 “MSG Ground Segment LRIT/HRIT Mission Specific Implementation” EUM/MSG/SPE/057; Issue 6; 21 June 2006
- [MTP] “The Meteosat Archive; Format Guide No. 1; Basic Imagery: OpenMTP Format”; EUM FG 1; Rev 2.1; April 2000
- [SGS] “MSG Ground Segment LRIT/HRIT Mission Specific Implementation”; EUM/MSG/SPE/057; Issue 6; 21 June 2006
- [refl] “Conversion from radiances to reflectances for SEVIRI warm channels” EUM/MET/TEN/12/0332
- [bt] “The Conversion from Effective Radiances to Equivalent Brightness Temperatures” EUM/MET/TEN/11/0569

m

- mipp, 9
- mipp.cfg, 10
- mipp.log, 10
- mipp.mda, 9
- mipp.xrit._xrit, 11
- mipp.xrit.bin_reader, 11
- mipp.xrit.convert, 12
- mipp.xrit.GOMS, 10
- mipp.xrit.loader, 11
- mipp.xrit.mda, 12
- mipp.xrit.MSG, 10
- mipp.xrit.MTP, 10
- mipp.xrit.sat, 12
- mipp.xrit.SGS, 11
- mipp.xsar.mda, 12
- mipp.xsar.sat, 12

C

CalibrationError, 9
ConfigReaderError, 9

D

debug_on() (in module mipp.log), 10
dec10216() (in module mipp.xrit.convert), 12
DecodeError, 9
decompress() (in module mipp.xrit._xrit), 11
dont_eval (mipp.mda.Metadata attribute), 9

E

emit() (mipp.log.NullHandler method), 10

G

get_logger() (in module mipp.log), 10

H

hrpt_dec10216() (in module mipp.xrit.convert), 12

I

ignore_attributes (mipp.mda.Metadata attribute), 9
ignore_attributes (mipp.xrit.mda.Metadata attribute), 12
ignore_attributes (mipp.xsar.mda.Metadata attribute), 12
ImageLoader (class in mipp.xrit.loader), 11

L

list() (in module mipp.xrit._xrit), 11
load() (in module mipp.xrit.sat), 12
load() (in module mipp.xsar.sat), 12
load_electrol() (in module mipp.xrit.sat), 12
load_files() (in module mipp.xrit.sat), 12
load_goes11() (in module mipp.xrit.sat), 12
load_goes12() (in module mipp.xrit.sat), 12
load_goes13() (in module mipp.xrit.sat), 12
load_himawari8() (in module mipp.xrit.sat), 12
load_meteosat07() (in module mipp.xrit.sat), 12
load_meteosat09() (in module mipp.xrit.sat), 12
load_mtsat1r() (in module mipp.xrit.sat), 12
load_mtsat2() (in module mipp.xrit.sat), 12
logging_off() (in module mipp.log), 10

logging_on() (in module mipp.log), 10

M

Metadata (class in mipp.mda), 9
Metadata (class in mipp.xrit.mda), 12
Metadata (class in mipp.xsar.mda), 12
mipp (module), 9
mipp.cfg (module), 10
mipp.log (module), 10
mipp.mda (module), 9
mipp.xrit._xrit (module), 11
mipp.xrit.bin_reader (module), 11
mipp.xrit.convert (module), 12
mipp.xrit.GOMS (module), 10
mipp.xrit.loader (module), 11
mipp.xrit.mda (module), 12
mipp.xrit.MSG (module), 10
mipp.xrit.MTP (module), 10
mipp.xrit.sat (module), 12
mipp.xrit.SGS (module), 11
mipp.xsar.mda (module), 12
mipp.xsar.sat (module), 12
MippError, 9
mslice() (in module mipp.mda), 9

N

NavigationError, 9
NoFiles, 9
NullHandler (class in mipp.log), 10

R

raw_slicing() (mipp.xrit.loader.ImageLoader method), 11
read() (mipp.mda.Metadata method), 9
read_cds_expanded_time() (in module mipp.xrit.bin_reader), 11
read_cds_time() (in module mipp.xrit.bin_reader), 11
read_config() (in module mipp.cfg), 10
read_cuc_time() (in module mipp.xrit.bin_reader), 11
read_epiheader() (in module mipp.xrit.GOMS), 10
read_epilogue() (in module mipp.xrit._xrit), 11
read_float4() (in module mipp.xrit.bin_reader), 11
read_float8() (in module mipp.xrit.bin_reader), 11

[read_gts_message\(\)](#) (in module mipp.xrit._xrit), [11](#)
[read_imagedata\(\)](#) (in module mipp.xrit._xrit), [11](#)
[read_int2\(\)](#) (in module mipp.xrit.bin_reader), [11](#)
[read_int4\(\)](#) (in module mipp.xrit.bin_reader), [11](#)
[read_metadata\(\)](#) (in module mipp.xrit.GOMS), [10](#)
[read_metadata\(\)](#) (in module mipp.xrit.MSG), [10](#)
[read_metadata\(\)](#) (in module mipp.xrit.MTP), [10](#)
[read_metadata\(\)](#) (in module mipp.xrit.SGS), [11](#)
[read_mpef\(\)](#) (in module mipp.xrit._xrit), [11](#)
[read_mpef_clm\(\)](#) (in module mipp.xrit._xrit), [11](#)
[read_proheader\(\)](#) (in module mipp.xrit.GOMS), [10](#)
[read_prologue\(\)](#) (in module mipp.xrit._xrit), [11](#)
[read_uint1\(\)](#) (in module mipp.xrit.bin_reader), [11](#)
[read_uint2\(\)](#) (in module mipp.xrit.bin_reader), [11](#)
[read_uint4\(\)](#) (in module mipp.xrit.bin_reader), [11](#)
[read_uint8\(\)](#) (in module mipp.xrit.bin_reader), [11](#)
[ReaderError](#), [9](#)

S

[save\(\)](#) (mipp.mda.Metadata method), [9](#)
[strptime\(\)](#) (in module mipp), [9](#)

T

[token](#) (mipp.mda.Metadata attribute), [9](#)
[token](#) (mipp.xrit.mda.Metadata attribute), [12](#)
[token](#) (mipp.xsar.mda.Metadata attribute), [12](#)

U

[UnknownSatellite](#), [9](#)