PAINTER.jl Documentation

Release 0.6.3

JuliaPAINTER

November 02, 2016

Contents

`PAINTER.jl` is a Julia implementation of PAINTER: Polychromatic opticAl INTErferometric Reconstruction software.

See Credits for references.

Contents:

CHAPTER 1

Getting Started

1.1 Installation

`PAINTER.jl` uses the following registered Julia packages:

- OptimPack.jl: the Julia interface to OptimPack for solving the phases proximal operator.
- OIFITS.jl: Julia support for OI-FITS (optical interferometry data format).
- NFFT.jl: Julia implementation of the Non-equidistant Fast Fourier Transform (NFFT).
- Wavelets.jl: A Julia package for fast wavelet transforms.
- HDF5.jl: for writing JLD ("Julia data") variables.

They will be *automaticaly* installed during `PAINTER.jl` installation. Note that they require development tools included for example for OSX in `Command Line Tools` and for ubuntu in the `build-essential` package.

To install `PAINTER.jl`, type from a Julia session the following commands:

```
Pkg.update()
Pkg.add("PAINTER")
```

and relax!

It is recommended to install `PyPlot.jl` to monitor the iterations of the algorithm when the number of wavelengths is small, e.g. < 30. See PyPlot.jl page.

1.2 Usage

To load the `PAINTER.jl` module, type from a Julia session:

```
using PAINTER
```

If `PyPlot` is installed, it will be automatically loaded.

Iteration steps of `PAINTER.jl` are parallelized. To use parallel computing, start Julia with `nprocs` local process and load the module on all process:

```
$ julia -p nprocs
julia> using PAINTER
```

1.3 Path to OptimPack options

The file optpckpt.jl located in source file of PAINTER contains all OptimPack parameters for the phases proximal operator.

- ls, scl, gat, grt, vt, memsize, mxvl, mxtr, stpmn, stpmx. See OptimPack for details.

Default values are:

```
ls=OptimPack.MoreThuenteLineSearch(ftol=1e-8,gtol=0.95)
scl=OptimPack.SCALING\_OREN\_SPEDICATO
gat=0
grt=0
vt=false
memsize=100
mxvl=1000
mxtr=1000
stpmn=1E-20
stpmx=1E+20
```

Functions

2.1 Main function

painter (...)

- `painter` is defined with two methods:
  - Full parameters definition. This method is generally used to initialize the algorithm:

```
OIDATA,PDATA = painter(Folder, nbitermax, nx, lambda_spat, lambda_spec, lambda_L1, epsilon,
```

  - Specific structures. This method allows to restart the algorithm, for example if the number of iterations is not sufficient (see variable `nbitermax+=100`).

```
OIDATA,PDATA = painter(OIDATA,PDATA, nbitermax, aff; plotfunction)
```

- `painter` returns 3 structures:

```
OIDATA,PDATA = painter(...)
```

where:

  - `OIDATA`: contains all OIFITS information and user defined parameters.
  - `PDATA`: contains all the variables and arrays modified during iterations.

2.2 Auxiliary functions

paintersave (*filename::String*, *PDATA::PAINTER_Data*, *OIDATA::PAINTER_Input*)

Saves the structures `OIDATA`, `PDATA` into `*.jld` julia data files. The prefix of these structures is added before the "filename" base when writing the output files. See HDF5 package for details on the format.

```
filename= "datafile.jld"
cd("~/path/to/saved/data") # move to a different directory if necessary
paintersave(filename,PDATA,OIDATA)
```

painterload (*filename::String*)

Loads the structures from `*.jld` files. The files to be loaded must start with **OIDATA_** and **PDATA_** prefixes, but the filename entered as an argument should not have a prefix, since they are internally added by this function. Therefore, the filename of `painterload` is compatible with the one of `paintersave`.

```
PDATA2,OIDATA2 = painterload(filename)
```

The current version of the save function doesn't save the pointer to the user defined plot function. To warmstart the algorithm, the user must call the `painter(...)` with the personalized plot function as argument otherwise the default plot function is used.

painterfitsexport (*filename::String*, *PDATA::PAINTER_Data*, *OIDATA::PAINTER_Input*)

Saves the relevant information from `PDATA` (output data cube and associated criteria, reconstructed complex visibilities,...) and from `OIDATA` (wavelengths, input reconstruction parameters,...) into a FITS file "filename", which possibly includes a full path. The resulting FITS file has three HDUs : "Primary" is the reconstructed image cube, "INFO" contains the reconstruction parameters and criteria, and "VIS" contains the complex visibilities of the reconstruction, with the associated wavelengths and (U,V) points.

```
filename = "~/path/to/saved/data/myfitsdata.fits"
painterfitsexport(filename,PDATA,OIDATA)
```

painterplotfct (*PDATA::PAINTER_Data*, *OIDATA::PAINTER_Input*)

It is recommended to monitor the iterations of the algorithm when the number of wavelength is small, e.g. < 30.

The default function computes the number of subplots as a function of the number of wavelength if `nw<30`. Its is automatically called if `PyPlot` is installed and `aff=true`.

- The first figure shows the per-channel estimates projected on the domain support. The axis are defined by the field of view with no limitation of the amplitude (colorbars are different for all images).
- The second figure shows the primal and dual residuals (`crit1` and `crit2`) as a function of the iteration.

mask (*nx::Int*, *param::Int*, *choice::String*)

Creates a binary mask of size nx^2:

```
    Mymask3D = mask(nx,param,choice)
```

- `choice` can be a square (default: `choice="square"`) or a disk (`choice="disk"`).
- `nx` is the size of the image.
- `param` is the radius of the disk or the half size of the square.

Variables and structures

If parameters are not set, default values are used. For example

```
OIDATA,PDATA =  painter()
```

calls `painter` with all default values.

3.1 Variables

These two variables cannot be included in `OIDATA` structure.

- `nbitermax`: number of ADMM iterations. Default: `1000`.
- `aff`: if `aff=true` plots are enabled using `PyPlot.jl`. Default: `false`.

3.2 Variables in `OIDATA` structure

The structure `OIDATA` contains all OIFITS information and user defined parameters.

Execution Variables:

- `admm`: if `admm=false` the function only initializes the structures. The function `painter` can be used after to iterate the ADMM algorithm. Default: `true`.
- `CountPlot`: draw plot at each `CountPlot` iterations. Default: `10`.
- `PlotFct`: is a user defined function which is called at each `CountPlot` iterations. This function must respect the input argument of `painterplotfct` function and must call `PyPlot`, see *Examples and demo* section. Default: `painterplotfct`.

Initialization and initial estimate:

- `autoinit`: if `autoinit=true` some parameters are automatically set or rescaled. Default: `true`.

The parameters which are automaticaly initialized are `alpha`, `beta`, `rho_y_xi` and `rho_y_gamma`. They corresponds to parameters related to proximal operator for squared visibilities and for phases differences. Regularization parameters `lambda_spat` and `lambda_spec` are rescaled to be invariant with user parameters. `lambda_spat` is divided by the number of pixels and `lambda_spec` by the number of wavelength. The total flux is also normalized to allow the use of almost predefined parameters. The initial estimate is rescaled by the flux of the data.

Data and image related variables:

- `Folder`: path to the folder containing OIFITS/FITS files. Default: `./OIFITS`. If `./OIFITS` does not exists `src/OIFITS` in `PAINTER.jl/` default installation folder, containing FITS files for the demo, is used.
- `indfile`: allows to chose the set of OIFITS/FITS files in `Folder` that will be processed. `indfile` is an `Array{Int64,1}` containing the indexes of the files in alphabetical order. Default: all files.
- `indwvl`: allows to chose the set of processed wavelengths. `indwvl` is an `Array{Int64,1}` containing the indexes of the wavelengths in increasing order. Default: all wavelengths.
- `nx`: image size in pixels (the size of the image is nx^2). Default: `64`.
- `FOV`: Field Of View of the reconstructed image in ArcSecond. Default: `40e-3`.
- `mask3D`: Binary mask defining the support constraint. `mask3D` can be:
  - a path to a FITS file,
  - an Array,
  - an empty Array (no constraint).

  `mask3D` can be set by the function `mask`. Default: no constraint.
- `xinit3D`: Initial estimate of the object or of the complex visibilities. `xinit3D` can be:
  - a path to a FITS files containing the object,
  - an Array containing the object,
  - and Array containing the complex visibilities.

  Default: centered Dirac functions at all wavelengths.
- `dptype` define the kind of matrix difference used to generate differential phase, can be parameterized by `dpprm`:
  - `"all"` the difference between the first wavelength and all others (1-2, 1-3, ...), see Eqs. 35
  - `"diag"` the difference between all consecutive wavelengths (1-2, 2-3, ...)
  - `"ref"` the same as `"all"` but with a reference channel defined by `dpprm`, the same as `"all"` if ‘‘dp-prm‘‘=1
  - `"frame"` the difference between wavelength are performed inside non overlapping window with a size `dpprm`
  - `"sliding"` the difference between wavelength are performed using a sliding window with a size `dpprm`

  Default: if not given the default matrix difference is `"all"`, for details about other methods see [3].

ADMM algorithm parameters:

- `alpha`: weight for squared visibilities modulus data fidelity term, see Eqs. 25, 31 in [1]. Default: `1`.
- `beta`: weight for phases (closures and differential) data fidelity term, see Eqs. 25,31 in [1]. Default: `1`.
- `lambda_spat`: Spatial regularization parameter, see Eqs. 29, 31 in [1]. Default: nx^{-2}.
- `lambda_spec`: Spectral regularization parameter, see Eqs. 29, 31 in [1]. Default: `1e-2`.
- `rho_y`: ADMM parameter for data fidelity,see Eqs. 35, 50-52 in [1]. Default: `10`.
- `rho_spat`: ADMM parameter for Spatial regularization, see Eqs. 25, 31 in [1]. Default: `1`, (`0` to disable).
- `rho_spec`: ADMM parameter for Spectral regularization, see Eqs. 42, 55 in [1]. Default: `1`, (`0` to disable).
- `rho_ps`: ADMM parameter for positivity constraint, see Eq. 47, 54 in [1]. Default: `1`, (`0` to disable).

[1] Schutz, A., Ferrari, A., Mary, D. Soulez, F., Thiébaut, E., Vannier, M. “PAINTER: a spatio-spectral image reconstruction algorithm for optical interferometry”. JOSA A. Vol. 31, Iss. 11, pp. 2356–2361, (2014). arXiv

Secondary or specific paramaters: The defaults values of these parameteres are tuned for the general cases. Nevertheless, the user may modified them for specific applications.

- `lambda_L1`: regularization parameter for an l_1 constraint on the image. l_1 constraint emphasizes sparsity of objects (e.g. stars field). Default: `0`.
- `Wvlt`: array of wavelets basis for spatial regularization, see [2]. See Wavelets.jl for definitions. Default: first 8 Daubechies wavelets and Haar wavelets. `Wvlt = [WT.db1, WT.db2, WT.db3, WT.db4, WT.db5, WT.db6, WT.db7, WT.db8, WT.haar]`.
- `epsilon`: Ridge/Tikhonov regularization parameter, see Eqs. 29, 31 in [1]. Default: `1e-6`.
- `eps1`: stopping criterium for primal residual in ADMM algorithm. Default: `1e-6`.
- `eps2`: stopping criterium for dual residual in ADMM algorithm. Default: `1e-6`.

3.3 Constant in `OIDATA` structure

The structure `OIDATA`: contains also constants related to the data and extracted from OIFITS files.

- `nb`: number of bases.
- `nw`: number of wavelength.
- `U`: the U spatial frequencies matrix.
- `V`: the V spatial frequencies matrix.
- `P`: squared visibilities Matrix.
- `W`: squared visibilities variance Matrix.
- `T3`: phases closure matrix.
- `T3err`: phases closure variance matrix.
- `DP`: differential phases vector.
- `DPerr`: differential phases variance vector.
- `Xi`: dictionary of phases difference Vector.
- `K`: dictionary of phases difference variance vector.

For matrices, the column index is associated to the wavelength index.

3.4 Variables in `PDATA` structure

Useful outputs in the structure `PDATA` are:

- `PDATA.x`: the reconstructed 3D images !
- `PDATA.w`: positivity and support constraint. These constraints can be applied to `PDATA.x` with `PDATA.x.*(PDATA.w.>0)`.
- `PDATA.Fx`: non uniform Fourier transform of the reconstructed 3D images.
- `PDATA.H`: dictionary of phases to phases differences sparse matrix.
- `PDATA.crit1`: the primal residual of the ADMM algorithm.

[2] Schutz, A., Ferrari, A., Mary, D., Thiébaut, E., Soulez, F. "Large scale 3D image reconstruction in optical interferometry". EUSIPCO, 2015, Nice. arXiv

- `PDATA.crit2`: the dual residual of the ADMM algorithm.
- `PDATA.ind`: number of iterations, useful to re-run algorithm.

3.5 References

CHAPTER 4

Examples and demo

4.1 Demo for impatients

`PAINTER.jl` contains a demo file `painterdemo.jl` with an OIFITS folder in the default installation folder. To run the demo type:

```
using PAINTER
painterdemo()
```

`painterdemo()` run a simation with data generated with ASPRO with AMBER configuration and a gray object. The demo includes warm start, save and load of structures, a custom plot function (require PyPLot), ...

`painterdemo("gravity")` run simulation with data from the beauty contest 2016 (http://www.opticalinterferometry.com/beauty2016). Data will be downloaded in current folder and contains `gravity` simulation. In this case PAINTER uses the phases of the complexe visibities and the closure phases for the phases estimation. The demo includes save and load of structures, a custom plot function (require PyPLot), ...

`painterdemo("bc04")` run simulation with data from the beauty contest 2004. Data are monochromatic. The demo includes save and load of structures, a custom plot function (require PyPLot), ...

4.2 User parameters and single execution for `painterdemo()`

- The folowing parameters are set by the user:

```
    path        = '../OifitsFolder'
    FOV         = 0.01
    indwvl      = 1:30
    nx          = 64
    eps1        = 1e-4
    eps2        = 1e-4
    rho_y       = 10
    alpha       = 1e4
    beta        = 1e5
    rho_spat    = 4
    rho_ps      = rho_spat
    lambda_spat = 1e-5
    rho_spec    = 1/2
    lambda_spec = 1e-5
    dptype      = "sliding" # type of differential phases
    aff         = true      # plot is enabled
    nbitermax   = 100
```

PAINTER.jl will extract OIFITS informations from all files in the folder ../OifitsFolder and will restrict the analysis to the first 29 wavelengths.

- The initial estimate is the default. ADMM is enabled by default and will run the algorithm for 100 iterations.
- The support constraint is defined by a disk:

```
mask3D = mask(nx, int(nx/2 -3), choice="disk")
```

- Other parameters take the default values.

PAINTER.jl is then executed:

```
OIDATA, PDATA = painter(Folder=Folder, nbitermax=nbitermax, nx=nx, lambda_spat=lambda_spat=lambda_spa
```

4.3 Algorithm warm start

PDATA contains all variables and array modified during iterations, including the Lagrange multipliers. This allows a warm start of the ADMM algorithm. This is useful for example when the iterations have been stopped by nbitermax but the algorithm has not yet converged.

In this example the user wants 1000 additional iterations with disabled plots:

```
nbitermax += 1000
aff = false
OIDATA, PDATA_new = painter(OIDATA,PDATA, nbitermax, aff, PlotFct = Plotfunction)
```

PDATA_new is used to store the new auxiliary variables.

4.4 Outer iterations mode

It is possible to save the estimates (or other variables) at each iteration using single iterations in a loop:

```
for n = 1:10
  nbitermax += 1
  OIDATA, PDATA = painter(OIDATA, PDATA, nbitermax, aff)
  saveX[n] = PDATA.x
  saveW[n] = PDATA.w
end
```

Note that this is a very time consuming process.

4.5 User defined plot function

It is possible to plot or to print some informations on available data during iterations. If PyPlot.jl is installed, painter will execute each CountPlot iterations the function defined by the variable PlotFct. This user defined function must respect the input arguments of painterplotfct:

Plotfunction (*PDATA::PAINTER_Data*, *OIDATA::PAINTER_Input*)

For example, to plot at each iteration the sum over all wavelengths of an estimated polychromatic object, projected on a support constraint:

```
using PyPlot

function Plotfunction(PDATA::PAINTER_Data,OIDATA::PAINTER_Input)
        x = PDATA.x
        s = (PDATA.w.>0.0)
        im2show = squeeze(sum(x.*s,3),3)
        imshow(im2show)
end

OIDATA,PDATA = painter(..., PlotFct = Plotfunction)
```

Credits

PAINTER.jl is a julia implementation of *PAINTER: Polychromatic opticAl INTErferometric Reconstruction* algorithm described in [1] and [2].

PAINTER.jl was developed at:

> Laboratoire J.-L. Lagrange
> Université de Nice Sophia-Antipolis, CNRS,
> Observatoire de la Côte d'Azur,

by Antony Schutz and André Ferrari.

The development of PAINTER.jl was partially supported by the POLCA project funded by the French Agence Nationale pour la Recherche (ref. ANR-10-BLAN-0511).

5.1 License

PAINTER.jl is released under under the MIT "Expat" License.

5.2 References

The PAINTER algorithm is described in [1]. The original MATLAB code is available here but the use of PAINTER.jl is highly recommended. PAINTER.jl implements an accelerated version of PAINTER described in [2].

[1] Schutz, A., Ferrari, A., Mary, D. Soulez, F., Thiébaut, E., Vannier, M. "PAINTER: a spatio-spectral image reconstruction algorithm for optical interferometry". JOSA A. Vol. 31, Iss. 11, pp. 2356–2361, (2014). arXiv

[2] Schutz, A., Ferrari, A., Mary, D., Thiébaut, E., Soulez, F. "Large scale 3D image reconstruction in optical interferometry". EUSIPCO 2015, Nice. arXiv

M

P