# Contents

1 Features 3
2 Quick setup 5
3 User Guide 7
   3.1 Installation .................................................. 7
   3.2 Getting Started with pypher ................................. 7
   3.3 Convenience scripts ......................................... 9
   3.4 Contributing .................................................. 10
4 Acknowledging 11
Compute an homogenization kernel between two PSFs

This code is well suited for PSF matching applications in both an astronomical or microscopy context.

It has been developed as part of the ESA Euclid mission and is currently being used for multi-band photometric studies of HST (visible) and Herschel (IR) data.
1. **Warp** (rotation + resampling) the PSF images (if necessary).

2. **Filter** images in Fourier space using a regularized Wiener filter,

3. **Produce** a homogenization kernel.

**Note:** *pypher* needs the pixel scale information to be present in the FITS files. If not, use the provided `addpixscl` method to add this missing info.

**Warning:** This code **does not**
- interpolate NaN values (replaced by 0 instead),
- center PSF images,
- minimize the kernel size.
CHAPTER 2

Quick setup

In most cases the code can be installed via

```
$ pip install pypher
```

and a kernel can then be produced from two PSFs with a simple command line

```
$ pypher psf_a.fits psf_b.fits kernel_a_to_b.fits
```

Other installation procedures are described on the installation page and further command line options here.
3.1 Installation

PyPHER works both with Python 2.7 and 3.4 or later

3.1.1 Option 1: Pip

$ pip install pypher

3.1.2 Option 2: from source

$ git clone https://git.ias.u-psud.fr/aboucaud/pypher.git
$ cd pypher
$ python setup.py install

3.1.3 Dependencies

pypher needs the following Python libraries to be installed:

- numpy (>=1.7.2)
- scipy (>=0.9.0)
- astropy (>=0.4)

In case these are not automatically installed using the pypi install procedure, either install them manually or use the requirements.txt file provided with the source install and simply:

$ pip install -r requirements.txt

3.2 Getting Started with pypher

3.2.1 Usage
$ pypher psf_source psf_target output
    [-s ANGLE_SOURCE] [-t ANGLE_TARGET] [-r REG_FACT]
$ pypher (-h | --help)

### Arguments

- **psf_source** (*str*) path to the high resolution PSF image (FITS file)
- **psf_target** (*str*) path to the low resolution PSF image (FITS file)
- **output** (*str*) output filename

### Options

- **-h, --help** print help
- **-r, --reg_fact** (*float*) regularization factor (default 1.e-4)
- **-s, --angle_source** (*float*) rotation angle in degrees to apply to psf_source (default 0.0)
- **-t, --angle_target** (*float*) rotation angle in degrees to apply to psf_target (default 0.0)

#### 3.2.2 Examples

The simplest example is the following

```bash
$ pypher psf_a.fits psf_b.fits kernel_a_to_b.fits
```

This will create, using the default options values, the kernel `kernel_a_to_b.fits` and a short log `kernel_a_to_b.log` with information about the processing.

A more complex call would look like

```bash
$ pypher psf_a.fits psf_b.fits kernel_a_to_b.fits -r 1.e-5 -s 27.45 -t 221.08
```

where the source and target angles are defined following the bottom figure.

#### 3.2.3 Regularization parameter

Deconvolution of an image being an ill-posed problem, we stabilize the solution by using *regularization*. We chose to use a Wiener filter with a $\ell_2$ penalization which imposes a relative degree of smoothness between neighboring pixels. If the image can be written as

$$ y = Hx + n, $$

the Wiener filter can be expressed as

$$ \frac{H^t}{|H|^2 + \lambda |D|^2} $$

where $H$ and $D$ are, respectively, the convolution and the differential operators, and $\lambda$ is the regularization factor. This parameter creates a compromise between the fidelity to the data and the smoothness, and must therefore be tuned.

The **optimal value** for $\lambda$ (*--reg_fact*) is the **signal-to-noise ratio** $S/N$ of the image being deconvolved, *i.e.* the source image.
3.2.4 Angle option

Because astronomical images may not be have been observed with the same orientation as the one of the PSF image, a rotation can be needed in order to create a kernel that is aligned with both the source image and target image orientation.

For this purpose, two optional parameters \(-s, --angle_source\) and \(-t, --angle_target\) can be provided to the code so it can rotate the PSFs before computing the kernel.

These angles are defined in a clockwise order, as shown in the figure below.

![Figure 3.1: Schematic of an image and its associated PSF with the reference direction of the telescope +Z and the angle definition.](image)

**Warning:** +Z is not to be mistaken with neither the rotation angle nor the scanning angle

3.3 Convenience scripts

3.3.1 addpixscl

Write the pixel scale in FITS file headers

```
$ addpixscl fits_files pixel_scale [--ext EXT]
$ addpixscl (-h | --help)
```

**Arguments**

- **fits_files** *(str or list of str)* name of FITS file(s)
- **pixel_scale** *(float)* pixel scale value in arcseconds
Options

- `h, --help` print help
- `e, --ext (int)` FITS extension number (default 0)

Examples

This script works on single files

```bash
$ addpixscl psf_a.fits 0.1
```

as well as on a list of files

```bash
$ addpixscl psf*.fits 0.3 --ext 1
```

3.4 Contributing

We welcome all contributions.

To report any bugs or file feature requests, please use the issue tracker. Otherwise feel free to fork the repository, make your own changes and send a pull request.
Acknowledging

If you make use of any product of this code in a scientific publication, please consider acknowledging the work by citing the following paper:

Boucaud et al. (2016) “Convolution kernels for multi-wavelength imaging”

Note: submitted to Astronomy & Astrophysics journal

Author  Alexandre Boucaud
Email    alexandre.boucaud [at] ias.u-psud.fr
Licence  This work is licensed under a 3-clause BSD license