pixsrc user's manual

Amitpal S. Tagore

June 6, 2017

Contents

1 What is pixsrc?

pixsrc is software for de-lensing objects that have been strongly gravitationally lensed. Among other things, users of pixsrc can:

- Reconstruct a pixelated source on two different irregular grids.
- Use a combination of functional forms, like a Sérsic profile, to model the source.
- Expand the source's surface brightness analytically using shapelets.
- Place a variety of priors on the source in a Bayesian framework.
- Use a number of pre-source-reconstruction tests to quickly reject bad lens models.
- Use CUDA to enable GPU computing.
- Add lens potential perturbations.
- Model interferometric data in the uv-plane.
- Compute and/or take into account interpolation errors.
- Simultaneously analyze multiple data sets of multiple lensed sources or images in different photometric/radio bands.

2 Installation

Setup environment to recognize shared dependencies and run install script.

3 Quick start

To get started quickly, the easiest way is to modify one of the pixsrc examples. Or, you can have pixsrc create a blank template. From the Unix prompt, type

unixprompt$ lensmodel pixsrc help template

This will create the "pixsrc_in" directory and several input files, which you can then modify. The basename will be "bname" and the imagename will be "iname".

4 Input files

4.1 A few important notes

All input files should be placed in the directory "pixsrc_in".

When turning pixsrc on, using the pixsrc on command in lensmodel, you must specify a basename. The basename tells pixsrc which files to read in. Below, any occurence of <basename> refers to this basename. You must also supply one or more FITS files for pixsrc to read in. Below, any occurrence of <imagename> refers to the name of the FITS file without the ".fits" extension. For example, <imagename> corresponds to the file imagename.fits.

Any input files that end with the ".reg" extension are DS9 region files. In order for pixsrc to understand these files, they must be saved image coordinates or WCS coordinates. If WCS, then the coordinates should be in degrees and not sexagesimal (hr:min:sec) format. This means that there should be a line in the region file that contains the word "image" or "fk5". Furthermore, all regions should be the polygon region, as opposed to the circle or box region. There can be multiple polygons and the polygons can self-intersect. The one exception is the source mask described below, which only accepts circle regions. There can be multiple circle regions.

4.2 Input files

<basename>.include
This include file lists the names of the FITS file(s) to be included in the lensing analysis. A cleaned image must be given for uv-plane modelling as well (to estimate boundaries and source/image scales). One file per line.

<imagename>.fits
This data file is a data file that contains the lensed images of the background source(s). There can be multiple data files, which should all be listed in the include file.

<imagename>.psf.fits
This PSF file contains the psf for the corresponding <imagename>.fits file. If the psf option (see parameters) is set to "1", then this file must exist.

<basename>.parameters
This parameters file contains all the parameters for pixsrcto read in. Please try pixsrc help parms or pixsrc help parms more for details about input parameters.

<basename>_<imagename>.datamask.reg
This data mask file masks pixels that are to be used in the lensing analysis. Other pixels may also be included in the lensing analysis if the sisterpix option (see parameters) is set.

<basename>_<imagename>.badpixelmask.reg
This bad pixel mask file masks pixels that are flagged as bad and not to be used in the lensing analysis. These pixels may, however, be used if the fillbadpix option (see parameters) is set. They will also always be used in some penalty functions (see parameters) which do not use the surface brightness values. Bad pixels may also be specified in a text file, **<basename>_<imagename>.badpixelmask.ascii**, where the first column is x coordinates and the second column is y coordinates. The pixels are numbered such that the bottom left pixel is (0,0). This feature is useful if you want to exclude all pixels below a certain surface brightness. (You would have to make the pixel list yourself.)

<basename>_<imagename>.chi2mask.reg
This χ^2 mask file masks pixels where the χ^2 term should be calculated. No other pixels will be used to compute the χ^2 term. However, if the Bayesian evidence is the primary statistic being computed, then all eligible pixels will be used to constrain the source surface brightness. Note: the value of the χ^2 term can be monitored in the output log file.

<basename>_<imagename>.mmimages.*.reg
This mmimages file masks individual lensed images of a single background source. If there is more than one source being lensed, then multiple such files can exist. The asterisk "*"in the filename can be anything and is used to distinguish background sources. Penalty functions (see parameters) use the mmimages files to compute penalties. Thus, if a penalty function is turned on, then at least one mmimages file must exist. An mmimages file must contain at least two polygon regions.

<basename>_<imagename>.srcmask.reg
This source mask file masks regions of the sky within which the source grid lies. No source pixel lies outside this region. Unlike the other region files, the source mask only accepts circle regions; no polygons allowed.

<analytic source file>
This analytic source file contains analytic source models. The filename is specified by the src option (see parameters). There can be multiple source models. If there are N source models, then the first N lines specify the models. These lines must contain nine entries each: model I_0 ra

dec e PA R Z n. model specifies the functional form of the source; options are "none", "sersic", and "vector_<vecfilename>". "none" specifies a source with zero surface brightness. "sersic" specifies a Sérsic profile. The surface brightness, $I(r)$, along the major axis is given by: $I(r) = I_0 \exp[-(r/R)^{1/n}]$, where r is the distance along the major axis. e denotes the ellipticity and PA the position angle (in degrees) of the major axis, measured east of north. The Z parameter is unused, and positions (ra and dec) are in units of arcseconds. "vector_<vecfilename>" allows an arbitrary surface brightness distribution to be lensed. The file <vecfilename> contains 3 columns: right ascension, declination, surface brightness, where the positions are given in arcseconds offsets, using the coordinate system defined using "coorsys:". The syntax is: vector_<vecfilename> I_0 ra dec m11 m12 m21 m22 Z. I_0 is an overall scaling of the surface brightness. ra and dec are position offsets in arcseconds. Z is ignored. m?? denote elements of a linear transformation of the coordinates, so that the new coordinates are given by

$$\begin{pmatrix} ra' \\ dec' \end{pmatrix} = \begin{pmatrix} m11 & m12 \\ m21 & m22 \end{pmatrix} \begin{pmatrix} ra \\ dec \end{pmatrix} \tag{1}$$

This allows for rotations, stretches, reflections, etc. of the coordinates to be optimized. Then, the N+1 through 2N lines specify which parameters are to be varied. Each of these lines must contain 8 entries. "0" indicates the parameter is fixed, and "1" indicates the parameter is to be varied.

<analytic source bounds file>

This analytic source bounds file contains bounds on analytic source model parameters. There can be multiple bounds. Each bound is a linear combination of source model parameters. The sum of the combination must fall within the lower and upper limits to avoid a very large penalty. The file must have the following form: isrc iparm ai jsrc jparm aj ... lo hi where isrc is the source number (starting from 1), iparm is the parameter number (starting from 1), ai is the weight of the parameter, lo is the lower bound, and hi is the upper bound. Thus, each bound has the form: lo <= ai*pi + aj*pj + ... + aN*pN <= hi where pj=p[jsrc][jparm] and so forth, and any source model that violates the bound is penalized heavily. So, if there are N terms in a bound, then that line in the file must have 3N+2 terms. This file is optional.

<analytic source step sizes file>

This analytic source step sizes file contains step sizes for setting up the initial simplex for source model optimization. The number of lines in the file must equal the number of sources in the model. There must be eight entries on each line. Each entry specifies the step size for the corresonding parameter. This file is optional. Default step sizes are: 1 1 1 0.25 45 1 0 3 Position step sizes are in units of arcseconds.

For completeness, there is an input file associated with non-parametric lens potential perturbations. However, this is still under refinement and is not discussed here. ADD SHAPEINT MASK AS WELL.

5 Parameters file

5.1 General

debug:

Turn debug mode on(1) or off(0).

Debug mode will print lots of internal variables to file. It can be slow and require lots of disk space.

debug: 1

Turn debug mode on.

debug: 0

Turn debug mode off.

default: 0

fatalwarn:

Turn fatal warnings during initialization on(1) or off(0).

Turn fatal warnings during initialization on(1) or off(0).

fatalwarn: 1

Turn fatal warnings on.

fatalwarn: 0

Turn fatal warnings off.

default: 1

coorsys:

Set coordinate system and reference point.

The positions of lenses in the lens model are set using offsets in right ascension and declination, measured in arcseconds. This parameter defines an origin with respect to which these offsets are applied. IMPORTANT: When specifying lens models, angles are measured west of north.

coorsys: <SYS> <A> <B> <C> <D>

SYS is the coordinate system you want to use and can be either J2000, B1950, ecliptic (2000), galactic (2000). For ecliptic or galactic do not type "(2000)". A is the right ascension (RA) of the reference position (degrees). B is the declination (Dec) of the reference position (degrees). C is the RA offset of the reference position (arcseconds). D is the Dec offset of the reference position (arcseconds).

default: none

psf:

Specify the point spread function (PSF).

The PSF is computed only once when pixsrc is initialized. During convolution, less than 1% of the flux is lost. Input PSF files must be square with odd dimensions.

psf: 0

Turn off PSF convolution.

psf: 1

Read PSF from file named "pixsrc_in/<imagename>.psf.fits".

psf: 2 <A> <B> <C> [D]

Set PSF as an elliptical Gaussian. <A> is the FWHM along the major axis (arcseconds). <B> is the FWHM along the minor axis (arcseconds). <C> is the position angle of the major axis (degrees, east of north). [D] is the oversampling factor of the PSF in x & y directions (default 101).

default: 0

oversample:

Specify the image pixel oversampling factor, for creating the lensing operator, in x & y directions.

oversample: <value>

Set oversampling factor to <value>.

default: 1

noise:

Set the 1-sigma of Gaussian noise in the data.

noise: <value>

Set the 1-sigma of Gaussian noise in the data to <value>.

default: 1

statistic:

Select which statistic will be computed for lens model ranking.

Any additional penalty functions (see penalty? commands) will be added to this statistic.

statistic: 0

Compute Bayesian evidence.

statistic: 1

Compute traditional χ^2: $[(data - model)/sigma]^2$.

statistic: 2

Compute nothing.

default: 0

magnification:

Turn magnification calculation on(1) or off(0).

magnification: 1

Compute magnification.

magnification: 0

Do not compute magnification.

default: 0

verbosity:

Control how often pixsrc prints to the screen.

Currently, "verbosity: 2" is identical to "verbosity: 3" and prints as much as possible.

verbosity: 1

Print very little.

verbosity: 2

Print only significant progress.

verbosity: 3

Print as much as possible.

default: 3

images:

Select which data and/or images will be written to file.

Files to be written include (but not limited to): data, models, residuals, source reconstructions.

images: 0

Write nothing to file.

images: 1

Write text files only.

images: 2

Write FITS files only.

images: 2

Write text and FITS files.

default: 3

varystrength:

Trace the lens model statistic and/or magnification as a function of regularization strength.

varystrength: <A> <B> <C>

Trace the statistic and/or magnification from <A> to <B>, taking <C> logarithmic steps.

default: 0 0 0

threads:

Set the number of CPU threads per input data file.

This parameter specifies the maximum number of CPU threads per input data file. It also specifies the maximum number of input data files to simultaneoulsy analyze. Thus, the maximum possible number of CPU threads is the square of this parameter. Some optimized, multithreaded basic linear algebra subroutines (BLAS) libraries will use some pre-configured number of threads, and pixsrc cannot control this.

threads: <A>

Set the maximum number of CPU threads per data file to A.

default: 8

details:

Write detailed log file for each pixsrc run: on(1), off(0).

details: 1

Write log file.

details: 0

Do not write log file.

default: 0

hacksrc:

Specify values of source parameters (as stored in code/C++ array).

Specify values of source parameters (as stored in code/C++ array). Not applicable to analytic source.

hacksrc: 0 [filename] hacksrc: 1 <filename>

Do not hack the source vector in the code. Do not solve for most probable source. Instead, use source parameters found in <filename>.

default: 0

dataname:

Set the name of the input data file to which subsequent commands will apply.

The argument to this command can be the entire filename (excluding directory prefixes) or the filename with the ".fits" extension omitted.

dataname: <name>

Set the name of the input data file to which subsequent commands apply to <name>.

default: none

rottrans:

Rotate and translate input data, relative to coordinate system specified by "coorsys".

This option is useful for registering multiple data files. For example, the following two sets of commands would produce the same results: "coorsys: J2000 160 25 0 0" and "rottrans: 0 0 0 0 0" "coorsys: J2000 160 25 1 1" and "rottrans: 0 0 0 -1 -1"

rottrans: <rot_ra> <rot_dec> <theta> <ra> <dec>

Rotate the data by an angle <theta> (given in degrees) about the point (<rot_ra>,<rot_dec>) and translate by <ra>,<dec> in R.A. and Dec. (All positions are given in arcsecond offsets.).

default: 0 0 0 0 0

noisemap:

Compute noise and S/N ratio maps in the source plane: on(1), off(0).

This can be a long and memory intensive process.

noisemap: 1

Compute noise and S/N ratio maps in source plane.

noisemap: 0

Do not compute noise and S/N ratio maps in source plane.

default: 1

outprecision:

Set the decimal precision for writing floating-point numbers to file.

outprecision: <value>

Set the decimal precision for writing floating-point numbers to file to <value>.

default: 7

fullmag:

Select whether model surface brightness across all image pixels or only those masked as good pixels will be used to compute magnification.

fullmag: 1

Use model surface brightness across all image pixels to compute magnification.

fullmag: 0

Use model surface brightness across only masked image pixels to compute magnification.

default: 1

pixelscalesrc:

Set the source pixel scale used to write FITS files.

The default value is 1/4 the image pixel scale.

pixelscalesrc: <value>

Set the source pixel scale used to write FITS files to <value>.

default: 1/4 img scale

lowmem:

Disabled – Use algorithms tuned for low memory systems: on(1), off(0).

Disabled.

lowmem: 0

Use normal algorithms.

lowmem: 1

Use low memory algorithms.

default: 0

5.2 Data and gridding

grid:

Specify grid on which the source will be reconstructed.

If shapelets are turned on, this option will be ignored. The fully adaptive grid is created by ray-tracing a subset of the data pixels to the source plane. pixel_skip specifies how many data pixels to ignore before ray-tracing a data pixel. An image plane pixel is ray-traced if (x*(imgY)+y)%skip_level==0, where x and y are the x- and y-coordinates of the pixel, and imgY is the number of pixels along the y-axis. For example, skip_level=2 selects every other pixel. For irregular Cartesian grids, the zeroth level grid is created using five pixels, spanning the source region. Smaller "nested" Cartesian grids are created if the magnification in those regions is sufficiently high. Because the size of the initial grid is arbritray, the magnification threshold for subgridding is also arbritrary. shift_level ascribes a magnification to the zeroth level grid. It must be between 1 and infinity, exclusive. In general, as shift_level increases, the number of source pixels decreases.

grid: 1 [pixel_skip]

Select the fully adaptive grid. pixel_skip controls which image pixels are used to create grid (default 2).

grid: 2 [shift_level]

Select irregular Cartesian grid. shift_level sets the reference magnification for the zeroth level grid (default 10).

default: 1 2

sisterpix:

Turn search for sister pixels on(1) or off(0).

The source region is mapped out using image pixels. Then, image pixels not flagged as bad pixels are ray-traced to the source plane. If the image pixel lies inside the source region or within one FWHM of the PSF, it is included in the analysis.

sisterpix: 1

Turn on search for sister pixels.

sisterpix: 0

Turn off search for sister pixels.

default: 0

fillbadpix:

Change bad pixels to good pixels and change the pixel value.

fillbadpix: 0 0

Do not change bad pixels.

fillbadpix: 1 noise

Change bad pixels and replace pixel values with noise.

fillbadpix: 1 <value>

Change bad pixels and replace pixel values with <value>.

default: 0 0

minangle:

Set the minimum angle in any triangle in the pseudo-Delaunay triangulation of the source plane grid.

Small angles can lead to problems in calculating derivatives and surface brightness.

minangle: <value>

Set the minimum angle to <value>, given in degrees.

default: 0.1

rmpix:

Remove source pixels not constrained by lensing operator: on(1), off(0).

Not removing unconstrained pixels may lead to numerical instability and/or more computational effort.

rmpix: 1

Remove unconstrained pixels.

rmpix: 0

Do not remove unconstrained pixels.

default: 1

raydirection:

Constrain image pixels by lensing source pixels forward (0) or by ray-tracing image pixels backwards (1).

As coded, constraining image pixels by lensing source pixels forward does not work very well.

raydirection: 1

Constrain image pixels by ray-tracing image pixels backwards.

raydirection: 0

Constrain image pixels by lensing source pixels forward.

default: 1

5.3 Shapelets

shapelets:

Use shapelets (instead of using a source grid) to parameterize source.

Although grid parameters are ignored, a sparse grid is constructed to compute the convex hull. Source regularization can still be used.

shapelets: 0 [nx] [ny]

Do not use shapelets.

shapelets: 1 <nx> <ny>

Use <nx> × <ny>shapelets to reconstruct the source.

default: 0 10 10

shapeparms:

Fix shapelets center and/or scale.

shapeparms: 0 [ra] [dec] [sc]

Automatically estimate source center and scale.

shapeparms: 1 <ra> <dec> [sc]

Fix shapelets center to (<ra>,<dec>), given as R.A. and Dec. offsets.

shapeparms: 2 [ra] [dec] <sc>

Fix shapelets scale to <sc>, given in arcseconds.

shapeparms: 3 <ra> <dec> <sc>

Fix shapelets center to (<ra>,<dec>), given as R.A. and Dec. offsets, and shapelets scale to <sc>, given in arcseconds.

default: 0

shapepixsplit:

Set minimum and maximum pixel splitting in image plane.

shapepixsplit: <min> <max>

Set minimum and maximum pixel splitting in image plane to <min> and <max>.

default: 1 8

5.4 Regularization

regorder:

Select form of regularization.

regorder: -1

Penalize large source sizes (only valid when using shapelets).

regorder: 0

Penalize sources with large surface brightnesses.

regorder: 1

Penalize sources with large gradients of surface brightness.

regorder: 2

Penalize sources with large Laplacians of surface brightness.

default: 2

regstrength:

Guess or fix source regularization strength.

A value of zero will turn regularization off. However, it is numerically more stable to fix the regularization strength to a small value. The order of magnitude of the small value will depend on the noise level in the data.

regstrength: -<value>

Fix regularization strength to <value>, where <value> is positive.

regstrength: <value>

Provide initial guess, where <value> is positive.

default: 1

regaccuracy:

Compute higher accuracy regularization: on(1), off(0).

Higher accuracy will require more computational effort. This option is only available for some combinations of grids and regularization. This option has no effect on shapelets, which are analytic and exact. For derivative-based regularization schemes, higher accuracy means using the divergence theorem to compute derivatives. Higher accuracy is not yet available for curvature (Laplacian-based) regularization. For analytic source regularization (ASR), higher accuracy means that all pixels connected to a given pixel are used in regularizing it, as opposed to only using the nearest four pixels.

regaccuracy: 1

Compute higher accuracy regularization.

regaccuracy: 0

Do no compute higher accuracy regularization.

default: 0

reg:

Turn regularization on(1) or off(0).

Turning regularization off is necessary to use strictly analytic source(s) to model the data. Otherwise, to turn regularization off, it is numerically more stable to fix the regularization strength to a small value. The order of magnitude of the small value will depend on the noise level in the data.

reg: 1

Regularize the source.

reg: 0

Do not regularize the source.

default: 1

optfinder:

Select how the regularization strength is found.

Maximizing the evidence can be faster; however, the root finder is more robust.

optfinder: 1

Maximize the evidence to find optimal regularization strength.

optfinder: 0

Use a derivative-based root finder.

default: 1

regftol:

Set convergence criterion for source regularization strength.

This parameter is passed to the GSL multidimensional minimizer if evidence is maximized. This parameter is the fractional change needed for convergence if using the derivative-based solver. See "optfinder" for more details about finding regularization strength.

regftol: <value>

Set the convergence criterion for source regularization strength to <value>.

default: 1e-3

5.5 Analytic source

src:

Fit one or more analytic sources to the data.

If regularization is turned on, analytic source regularization (ASR) will be used. If regularization is turned off, the analytic source(s) will be used to model the data. Three source profile types are implemented: "none", "sersic", and "vector_<vecfilename>". "none" denotes a source with zero surface brightness everywhere. "sersic" denotes a Sérsic light profile with spherically symmetric light profile, $I(r)$, given by $I(r) = I_0 \exp(-(r/R)^{1/n})$. If there are N sources, the first N lines of the file should each contain the source profile type, followed by 8 source parameters. The next N lines should contain 8 parameter flags which denote whether the corresponding source parameter will(1) or will not(0) be optimized. For the "sersic" profile type, the 8 source parameters correspond to: I_0, ra, dec, e, PA, R, Z, k; where I_0 is the normalization, ra and dec denote the source center in R.A. and Dec. offsets, e and PA are

the ellipticity and position angle (east of north) of the major axis, R is the scale radius, Z is an ignored parameter, and n is the Sérsic index. "vector_<vecfilename>" allows an arbritrary surface brightness distribution to be lensed. The file <vecfilename> contains 3 columns: right ascension, declination, surface brightness, where the positions are given in arcseconds offsets, using the coordinate system defined using "coorsys:". Like the "sersic" profile, there can be multiple sources. The syntax is: vector_<vecfilename> I_0 ra dec m11 m12 m21 m22 Z. I_0 is an overall scaling of the surface brightness. ra and dec are position offsets in arcseconds. Z is ignored. m?? denote elements of a linear transformation of the coordinates, so that the new coordinates are given by

$$\begin{pmatrix} ra' \\ dec' \end{pmatrix} = \begin{pmatrix} m11 & m12 \\ m21 & m22 \end{pmatrix} \begin{pmatrix} ra \\ dec \end{pmatrix} \tag{2}$$

This allows for rotations, stretches, reflections, etc. of the coordinates to be optimized.

src: <filename>

Read analytic source parameters from "pixsrc_in/<filename>".

default: none

srcbound:

Specify bounds for analytic source parameters.

A bound consists of a combination of linear terms and a lower and upper bound for the combination.. If there are N bounds, there are N lines. For a particular bound, if there are M terms, there are 3*M+2 entries per line: isrc iparm ai jsrc jparm aj lower_bound upper_bound, where isrc¿=1 and 1¡=iparm¡=8. The bound is given by: lower_bound <= ai*pi + aj*pj + ... + aN*pN <= upper_bound, where pj is parameter number jparm of source number jsrc.

srcbound: <filename>

Read analytic source bounds from "pixsrc_in/<filename>".

default: none

srcstepsize:

Specify initial stepsizes for analytic source parameters.

If there are N sources, there are N lines containing 8 entries each, and each entry specifies the stepsize for the corresponding analytic source parameter.

srcstepsize: <filename>

Read analytic source stepsizes from "pixsrc_in/<filename>".

default: none

srcrestart:

Set the number of optimizations of the analytic source parameters to perform.

Selecting the best of several optimizations helps ensure a global minimum is found.

srcrestart: <value>

Performs <value> optimizations of the analytic source parameters.

default: 1

srcexact:

Set method for lensing analytic source: exact(1), approx(0).

Exact method does not use lensing matrix but ray-traces to source plane directly. Approximate methos uses lensing matrix and source grid, which can be quicker. If using exact method the source will not appear in output source-plane images, and analytic source regularization will be disabled. If using approximate method, oversampling can still be used.

srcexact: 1

Do not use lensing matrix and source grid.

srcexact: 0

Use lensing matrix and source grid.

default: 0

srcftol:

Set convergence criterion for analytic source optimization.

This parameter is passed to the GSL multidimensional minimizer.

srcftol: <value>

Set the convergence criterion for analytic source parameters to <value>.

default: 1e-3

srcctrguess:

Estimate center of source for each lens model: on(1), off(0).

For each lens model, the average of the brightness-weighted positions of the ray-traced image-plane pixels is chosen as an initial guess for the source center.

srcctrguess: 1

Estimate center of source for each lens model.

srcctrguess: 0

Do not estimate center of source (use user-specified center).

default: 1

5.6 Uv plane modelling

uvdata:

Specify a file containing visibility measurements for uv-plane modelling.

The file containing visibility measurements have 4 columns: u, v, real_vis, imag_vis. The file containing integrated visibility measurements is produced by pixsrc. It has 6 columns: u, v, real_integral, imag_integral, real_noise, imag_noise.

uvdata: 0 [filename]

Do not do uv-plane modelling.

uvdata: 1 <filename>

Read visibility measurements from pixsrc_in/<filename>.

uvdata: 2 <filename>

Read integrated visibility measurements from pixsrc_in/<filename>.

default: 0

uvmodelpos:

Specify a file containing uv positions where model visibilities should be calculated.

This should be a binary file, which can be loaded into a C++ double array. First number is the number of uv points. The remaining numbers should be all u points, followed by all v points. This is valid if uvdata is turned on with option 1. This is a slow, non-optimized calculation and is only really intended to be performed on the final lens and source model.

uvmodelpos: 0 [filename]

Do not produce model visibilities.

uvmodelpos: 1 <filename>

Read uv positions from pixsrc_in/<filename>.

default: none

uvmatrixsize:

Specify dimensions of submatrices in "divide and conquer" multiplication.

Two matrices will be allocated in memory. Each contains 2×<dim1>×<dim2> double elements. The default setting requires approximately 8Gb of memory, unless the number of uv data points is less than <dim1> and/or the number of image pixels is less than <dim2>. It may be more efficient to keep <dim1> greater than <dim2>.

uvmatrixsize: <dim1> <dim2>

Process <dim1> complex uv data points and <dim2> image pixels at a time.

default: 100000 2500

uvpbeam:

Turn primary beam application on or off.

If primary beam is being applied, the code will look for "pixsrc_in/<basename>_<imagename>.pbeam.dat". If the file has only one non-blank line containing the alphanumeric string <descr>, then the primary beam found in "pixsrc_in/<imagename>.pbeam.<descr>.fits" will be applied to all uv data points. If the file is empty or does not exist, the primary beam found in "pixsrc_in/<imagename>.pbeam.fits" will be applied to all uv data points. The primary beam can, however, vary between baselines. To achieve this, the file should contain as many lines as there are complex uv data points. Each line should, as before, contain one alphanumeric string <descr>, which corresponds to the FITS file, "pixsrc_in/<imagename>.pbeam.<descr>.fits". NOTE: Baseline-specific primary beams are not yet implemented.

uvpbeam: 0

Do not apply primary beam.

uvpbeam: 1

Apply primary beam.

default: 0

uvpixelscale:

Set the image pixel scale for uv modelling.

uvpixelscale: <value>

If <value> is negative, the pixel scale is set to $1/b_{max} \times 206265\times$(-<value>), where b_{max} is the maximum baseline. Otherwise, the pixel scale is set to <value>.

default: -0.5

uvrbf:
Set RBF type and scale length.

uvrbf: <type> <scale>

Set the RBF type to Gaussian (type=0). Set the scale length to scale.

default: 0 1

uvpadzeros:
Assume pixels surrounding the max contain zero surface brightness and use them to constrain RBF weights.

uvpadzeros: 0

Do not use surrounding pixels.

uvpadzeros: 1

Use surrounding pixels.

default: 1

uvtaper:
Apply uv taper; i.e., increase noise at longer baselines.

uvtaper: 0

Do not apply taper

uvtaper: -1

Automatically determine taper scale.

uvtaper: <value>

Apply taper: exp[(b/<value>)^2], where b is baseline lensgth.

default: 0

uvcutoff:
Set lower and upper cutoffs for uv-plane data.

uvcutoff: <lower> <upper>

Set lower and upper cutoffs for uv-plane data to <lower> and <upper>.

default: none

5.7 Gpu coding

cuda:
Turn GPU computing on/off and/or benchmark GPU(s).

cuda: help

List the available GPUs detected by pixsrc/CUDA and their names.

cuda: benchmark

Test the GPUs and their speeds.

cuda: ignore <gpulist>

Ignore the GPUs in the white-space separated list, <gpulist>.

cuda: <gpuname>

Use the GPU called <gpuname>.

default: none

5.8 Penalties

penalty6:

Disabled.

default: 0 0 0 0 0 0

penalty1:

Penalize the brightness-weighted source size (square arcseconds) of the reconstructed source. The weighting function is the absolute value of the surface brightness.

A number of penalty functions (priors) can be placed on the source reconstruction. In mode 1 (see below), the penalty is an additional χ^2 term based on the difference between what is expected from the prior and what is calculated. This expectation is referred to as the best estimate. This additional χ^2 is added to the primary statistic being computed (χ^2, evidence, or neither). In mode 2, hard lower and upper limits on the prior are specified. If the calculated value is below the lower limit or above the upper limit, then pixsrc is terminated and the χ^2 term is computed and returned, without computing the primary statistic. Otherwise, the primary statistic is computed as usual. There is also a hybrid mode (mode 3) in which mode 1 operates if the calculated value is within the lower/upper limits and mode 2 operates otherwise. By default, a "penalty?" command will apply to all input data files and to all "mmimages" region files. To change this behavior, use the "dataname" and "penaltyname" commands. For all penalty functions, the following arguments apply: arg <A>: mode: 0(off), 1(best estimate), 2(lower/upper bounds), 3(1 & 2) arg <B>: best estimate arg <C>: uncertainty in best estimate arg <D>: lower limit arg <E>: upper limit arg <F>: uncertainty outside lower/upper

penalty1: <A> <B> <C> <D> <E> <F>

See notes on "penalty1" for usage.

default: 0 0 0 0 0 0

penalty2:

Penalize the magnification.

penalty2: <A> <B> <C> <D> <E> <F>

See notes on "penalty1" for usage.

default: 0 0 0 0 0 0

penalty3:

Penalize the axis ratio (major/minor) of the "mmimages" (computed before source reconstruction).

penalty3: <A> <B> <C> <D> <E> <F>

See notes on "penalty1" for usage.

default: 0 0 0 0 0 0

penalty4:
Penalize the fractional overlap of the "mmimages" in the source plane (computed before source reconstruction). If there is no overlap between two images, then a negative value is returned, whose magnitude is the minimum distance between the two (ray-traced) images.

penalty4: <A> <B> <C> <D> <E> <F>

See notes on "penalty1" for usage.

default: 0 0 0 0 0 0

penalty5:
Penalize the size (square arcseconds) of the union of all ray-traced "mmimages" in the source plane (computed before source reconstruction).

penalty5: <A> <B> <C> <D> <E> <F>

See notes on "penalty1" for usage.

default: 0 0 0 0 0 0

penaltyname:
Set the name of the "mmimages" region file to which subsequent "penalty?" calls will apply.

The argument to this command can be the entire filename (excluding directory prefixes) or just the string between "mmimages." and ".reg".

penaltyname: <name>

Set the name of the "mmimages" region file to which subsequent "penalty?" calls apply to <name>.

default: none

5.9 Lens potential perturbation

nonparamlens:
Use non-parametric lens potential perturbations.

FILL IN THIS SECTION AFTER INPUT FORMAT IS CLEANED UP.

nonparamlens: 0

Turn off potential perturbations.

default: 0

reglens:
Set the regularization strength for the penalty on lens potential perturbations.

The lens potential penalty function is the magnitude of the gradient of the convergence.

reglens: <value>

Set the lens potential perturbation regularization strength to <value>.

default: 0

nplstepsize:
Set the stepsize for lens potential perturbations.

The stepsize is passed to the GSL multidimensional minimizer.

nplstepsize: <value>

Set the lens potential perturbation stepsize to <value>.

default: 2

nplftol:
Set convergence criterion for lens potential optimization.

This parameter is passed to the GSL multidimensional minimizer.

nplftol: <value>

Set the convergence criterion for lens potential perturbation to <value>.

default: 1e-3

npltpsreg:
Set TPS regularization strength and stepsize.

Set TPS regularization strength and stepsize.

npltpsreg: <strength> <ss>

Set the regularization strength to $10^{<strength>}$ and the stepsize (of the log regularization strength) to <ss>.

default: 0 3

5.10 Interpolation errors

interperr:
Compute interpolation errors from an analytic function: on(¿0), off(0).

interperr: 0

Do not compute interpolation errors.

interperr: <value>

Compute interpolation errors, oversampling by a factor of <value> in x & y dimensions.

default: 0

irscheme:
Specify what to do with interpolation errors.

irscheme: 0

Add interpolation errors to noise covariance matrix.

irscheme: 1

"Correct" the lensed and PSF-smeared image-plane surface brightness.

default: 0

6 Output files

6.1 A few notes

All output files will be placed in the directory "pixsrc_out".

Files ending in ".VECTOR" contain three columns: x-pixel coordinate, y-pixel coordinate, value. The coordinates starts from zero, with x increasing rightwards and y increasing downwards, with respect to the input data image.

Files ending in ".fits" are FITS files.

Files ending in any other extension are detailed below.

Additionally, the general format for output files that are NOT specific to a particular image is: "<basename>_*". The general format for output files that are specific to a particular image is: "<basename>_<imagename>_<imagenumber>_*", where <imagenumber> is a number assigned to a particular pixsrc run (when pixsrc does multiple source reconstructions for one lens model).

Lastly, in debug mode, more filetypes that those described below will be written, but these are not discussed here.

6.2 Output files

<basename>_<imagename>_<imagenumber>_data.fits or
<basename>_<imagename>_<imagenumber>_data.VECTOR
The data pixsrc used to do the source reconstruction (except when modelling in uv-plane) after any modifications/masking.

<basename>_<imagename>_<imagenumber>_lensedmps.fits or
<basename>_<imagename>_<imagenumber>_lensedmps.VECTOR
The model that is compared to the data (except for uv-plane modelling).

<basename>_<imagename>_<imagenumber>_lensedmpsnobo.fits or
<basename>_<imagename>_<imagenumber>_lensedmpsnobo.VECTOR
The model before convolution with the PSF.

<basename>_<imagename>_<imagenumber>_residuals.fits or
<basename>_<imagename>_<imagenumber>_residuals.VECTOR
The model subtracted from the data.

<basename>_<imagename>_<imagenumber>_psf.fits or
<basename>_<imagename>_<imagenumber>_psf.VECTOR
The point spread function used for image convolution.

<basename>_<imagename>_<imagenumber>_mps.fits or
<basename>_<imagename>_<imagenumber>_mps.VECTOR
The reconstructed source. Note that the VECTOR file contains the grid coordinates used by pixsrc. The FITS file is the source interpolated onto a regular grid.

<basename>_<imagename>_<imagenumber>_noise.fits or
<basename>_<imagename>_<imagenumber>_noise.VECTOR
The noise in the source plane. Note that the VECTOR file contains the grid coordinates used by pixsrc. The FITS file is the noise interpolated onto a regular grid.

<basename>_<imagename>_<imagenumber>_s2n.fits or
<basename>_<imagename>_<imagenumber>_s2n.VECTOR
The signal-to-noise in the source plane. Note that the VECTOR file contains the grid coordinates used by pixsrc. The FITS file is the S/N interpolated onto a regular grid.

<basename>_<imagename>_<imagenumber>_details.dat
A log file with one line per pixsrc run. This is mainly useful to check for fatal errors or to inspect the various terms that go into calculating the Bayesian evidence.

7 Examples

There are several pixsrc examples provided that highlight difference features. In the directory named "pixsrc_examples", there are several subdirectories. Each of these contains a lensmodel script that ends in ".in", which can be used to run a pixsrc example. The true lens model for each case is the same: two singular isothermal spheres (with different Einstein radii), separated by approximately 0.54 arcseconds. Snapshots of some of the more important outputs from pixsrc are are shown below.

7.1 Simple example

In this case, the lens model is fixed to the true value, and a source reconstruction is done using the default values. The only required inputs to pixsrc, given in the pixsrc_in/simple.parameters file, describe the world coordinate system, the point spread function, and the noise level in the data.

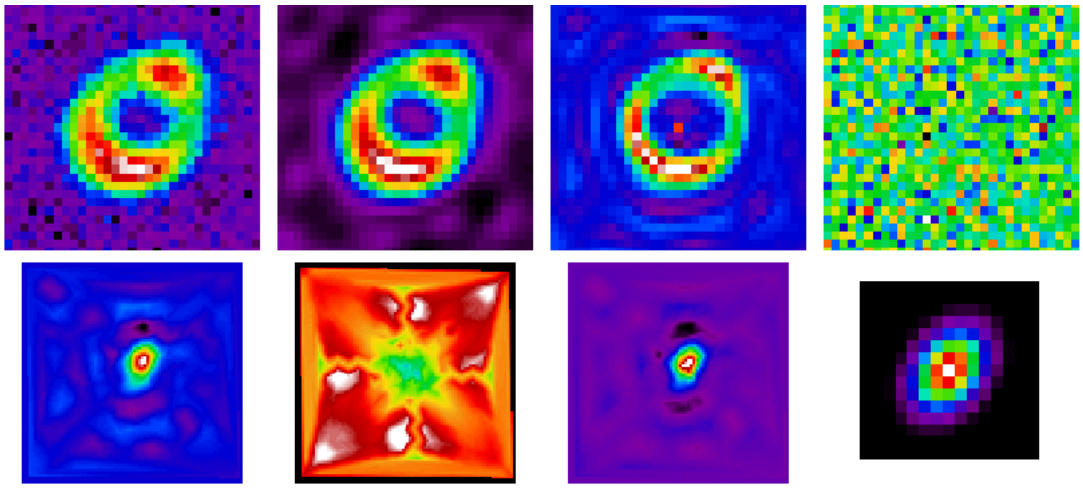

Figure 1: Top row, left to right: data, model, model before PSF convolution, residuals. Bottom row, left to right: reconstructed source, source-plane noise, source-plane signal-to-noise, PSF.

7.2 Typical (multiple datasets) example

In this example, two objects are modelled simultaneously using the same (true) lens model. Unlike the simpler example, some of the default pixsrc settings in the parameters file have been changed. The object in the top row of Fig. effig:typical is modelled using a parametric source model (an elliptical Sérsic profile), while the objetc in the bottom row is modelled using grid-free shapelets (a basis-set expansion of the source's surface brightness).

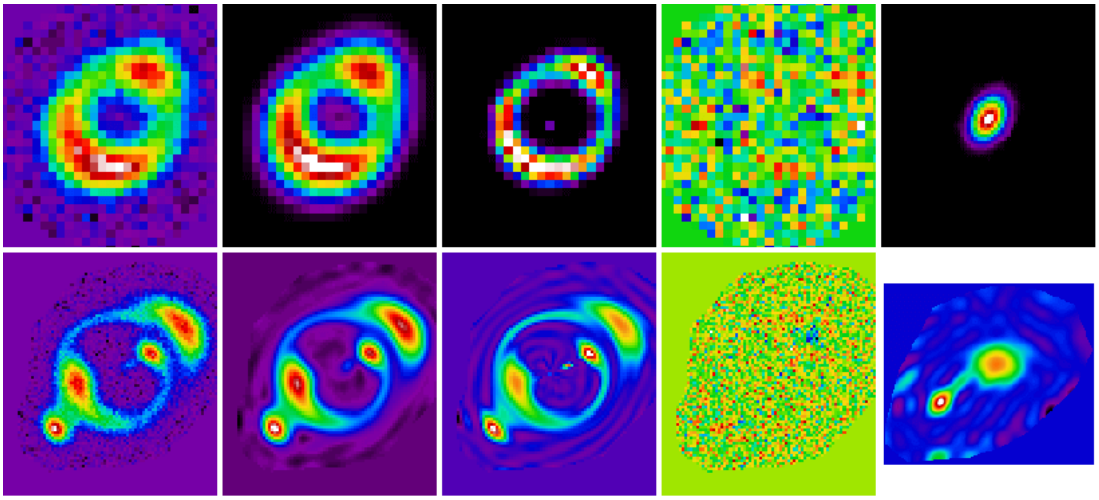

Figure 2: Top row, left to right: data, model, model before PSF convolution, residuals, reconstructed source. Bottom row, left to right: data, model, model before PSF convolution, residuals, reconstructed source.

7.3 Lens optimization example

This example shows how to efficiently minimize the lens model parameters. In an outer loop, the lens model parameters are varied, while in the inner pixsrc loop, the best source is found for each lens model. The example also demonstrates how to quickly reject bad lens models without having to do the more costly source reconstruction.

7.4 uv-plane example

This example shows how to model data taken with a radio interferometer, so that the complex visibility measurements are modeled directly. First in the analysis, an appropriate image-plane resolution is determined, which depends on the the longest baselines in data. Then, the image-plane is decomposed into radial basis functions, integrated, Fourier transformed, and evaluated at the positions of the visibility measurements. Finally, because of the large size of visibility datasets, a series of "divide-and-conquer" matrix algebra steps take place. The resutls of these steps are stored in binary files, which can be read in later to speed of future runs.

The example shows results of the uv-plane analysis on a very small dataset, but it also performs an image-plane analysis on the image-plane data used to create the visibility measurements. In this case, as can be seen from Fig. effig:uv, the image-plane residuals for the uv-plane reconstruction are not as good as those of the image-plane analysis, but this is largely due to the small size of the dataset used. Fig. effig:uvres, on the other hand, shows that the uv-plane residuals are consistent with the model fitting the data down to the noise level.

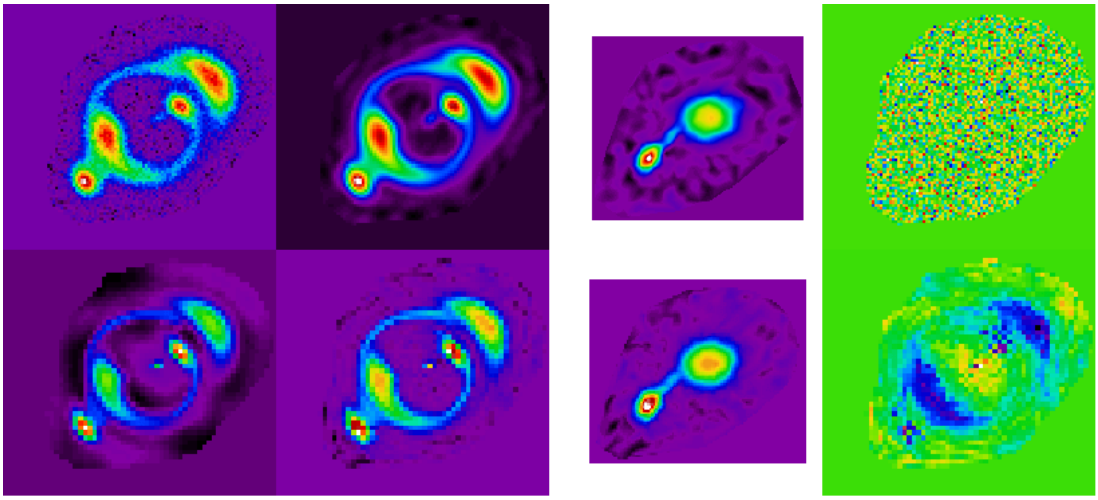

Figure 3: Top row (image-plane analysis), left to right: data, model, reconstructed source, residuals. Bottom row (uv-plane analysis), left to right: cleaned map, model, reconstructed source, residuals. Note that the model shown for the uv-plane analysis is not the model used by the code for constraining the visibility data. A similar statement holds for the residuals shown for the uv-plane analysis, as well. These are only shown here for comparison.

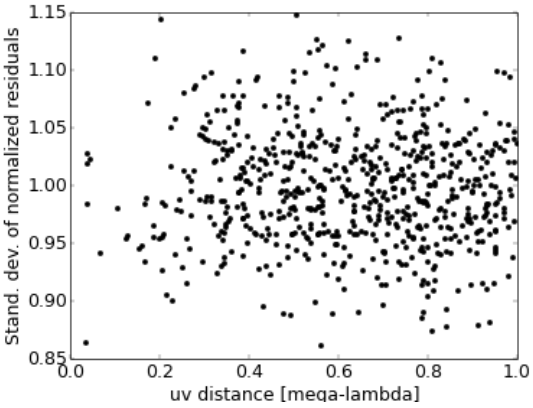

Figure 4: Standard deviation of normalized residuals ([data-model]/sigma). Each data point contains the standard deviation of 200 normalized model residuals.

7.5 Lens perturbation example

This example shows how to add non-parametric components on top of a parametric lens model. One of the two lensing galaxies in the true lens model is removed, and the code is able to recover the missing galaxy, as can be seen from Fig. effig:perturb and effig:pertkappa. Although this approach can be dangerous (in that it can produce unphysical features in the lens model), it can be useful to track down deficiencies in the lens model.

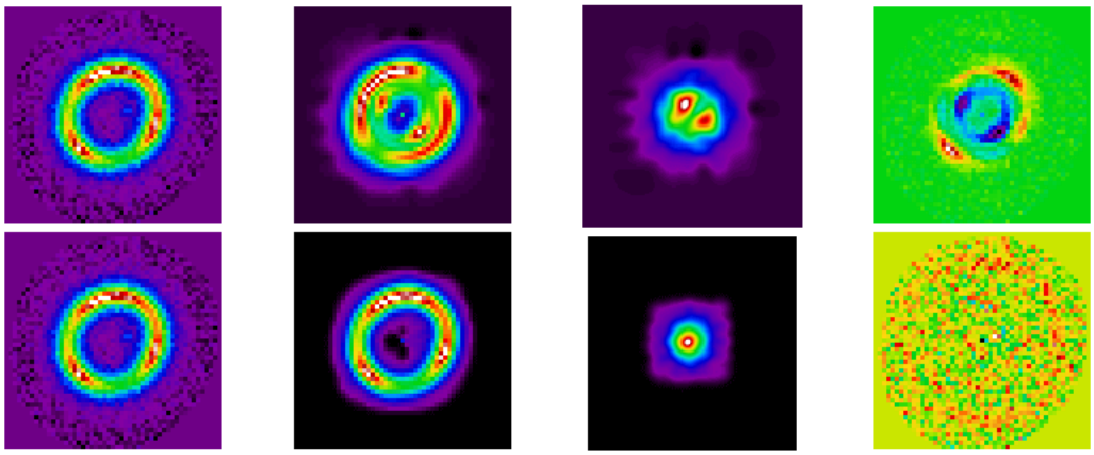

Figure 5: Top row (galaxy removed), left to right: data, model, reconstructed source, residuals. Bottom row (perturbations added), left to right: data, model, reconstructed source, residuals. Note that in this extreme case, the "perturbations" aren't really perturbations anymore; they represent a severe lacking in the lens model. However, this approach can still be used to expose something missing in the model, as long as the "perturbations" are physically plausible.

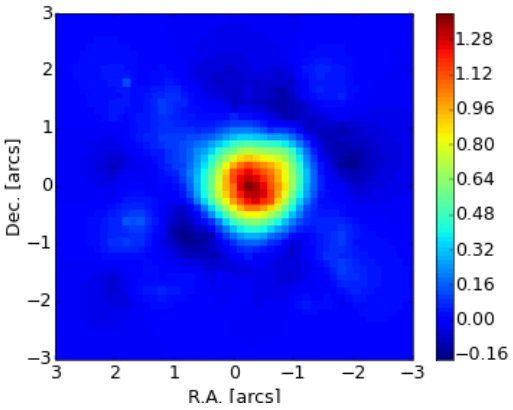

Figure 6: Twice the convergence map derived from the lens potential perturbations.