

lenstronomy: multi-purpose gravitational lens modelling software package

Simon Birrer*

*Department of Physics and Astronomy, University of California, Los Angeles, 475
Portola Plaza, Los Angeles, CA 90095-1547, USA*

Adam Amara

*Institute of Particle Physics and Astrophysics, Department of Physics, ETH Zurich,
Wolfgang-Pauli-Strasse 27, 8093, Zurich, Switzerland*

Abstract

We present `lenstronomy`, a multi-purpose open-source gravitational lens modeling python package. `lenstronomy` is able to reconstruct the lens mass and surface brightness distributions of strong lensing systems using forward modelling. `lenstronomy` supports a wide range of analytic lens and light models in arbitrary combination. The software is also able to reconstruct complex extended sources (Birrer et. al 2015) as well as being able to model point sources. We designed `lenstronomy` to be stable, flexible and numerically accurate, with a clear user interface that could be deployed across different platforms. Throughout its development, we have actively used `lenstronomy` to make several measurements including deriving constraints on dark matter properties in strong lenses, measuring the expansion history of the universe with time-delay cosmography, measuring cosmic shear with Einstein rings and decomposing quasar and host galaxy light. The software is distributed under the MIT license. The documentation, starter guide, example notebooks, source code and installation guidelines can be found at <https://lenstronomy.readthedocs.io>.

Keywords: gravitational lensing, software, image simulations

*corresponding author and lead developer

Email addresses: sibirrer@astro.ucla.edu (Simon Birrer),
adam.amara@phys.ethz.ch (Adam Amara)

1. Introduction

Strong gravitational lensing, the bending of light by foreground masses to such an extent that multiple images of the same source are formed, is an important phenomenon that can be used to probe the matter distribution and geometry of the universe. The detailed reconstruction of the light paths can be used to test the nature of the unknown components dark matter and dark energy. These dominate the matter-energy content of the Universe today.

In strong lensing studies, significant progress has been made in recent years in both quantifying the small scale matter distribution, with techniques such as gravitational imaging Vegetti et al. (2010, 2012); Hezaveh et al. (2016); Birrer et al. (2017a); Vegetti et al. (2018) and flux ratio anomalies Mao and Schneider (1998); Metcalf and Madau (2001); Dalal and Kochanek (2002); Nierenberg et al. (2014); Xu et al. (2015); Nierenberg et al. (2017), and measuring the expansion history of the universe, with time-delay cosmography Refsdal (1964); Schechter et al. (1997); Treu and Koopmans (2002); Suyu et al. (2010, 2014); Birrer et al. (2016); Bonvin et al. (2017). These successes have in part been made possible due to the development of state-of-the-art lens modelling software and algorithms that can extract the required lensing information from high resolution imaging data. Several of the codes used for doing this have been made publicly available to the community (see Appendix A).

From the current and upcoming surveys, the sample of known strong lenses is rapidly increasing (see e.g. Agnello et al., 2015; Nord et al., 2016; Schechter et al., 2017; Lin et al., 2017; Jacobs et al., 2017; Ostrovski et al., 2017; Williams et al., 2017; Lemon et al., 2018) and Treu et al. (submitted). This enlarged sample enables competitive measurements of the Hubble constant (see e.g. Treu and Marshall, 2016; Shajib et al., 2018; Suyu et al., 2018) as well as can strengthen constraints on dark matter properties on sub-galactic scales (see e.g. forecast by Gilman et al., 2017). To fully exploit the science potential of these new strong lensing data, our modelling tools need to be continually developed and improved.

In this publication, we present the first public release of `lenstronomy`, which is an open source multi-purpose strong lens modelling software package. `lenstronomy` has been used as a research tool throughout its devel-

opment. This includes being used for time-delay cosmography (Birrer et al., 2016), lensing substructure analysis (Birrer et al., 2017a), line-of-sight shear measurements from an Einstein ring (Birrer et al., 2017b) and its forecast to measure cosmic shear with Einstein rings (Birrer et al., 2018). Each of these tasks required the modelling of *Hubble Space Telescope* (HST) imaging data at the pixel level.

We developed `lenstronomy` to be stable, flexible and numerically accurate, with a clear API and user interface that could be used across different platforms. The `lenstronomy` software architecture was designed to be able to scale from the current era, where individual lenses are studied in detail, to the case where several hundreds of lenses, from future surveys, will need to be processed.

Given that an essential part of precision cosmology is the control of systematic errors, it is important for the community to develop multiple independent pipelines. This allows for the cross-checks that are necessary for complex precision measurements. Community standard benchmark efforts also make important contributions. For instance, the time-delay lens modelling challenge (Ding et al., 2018) offers a realistic and blind comparison framework in the domain of cosmography. Similar efforts are underway by the substructure lensing community. The public release of `lenstronomy` enables a transparent and effective comparison with other software used in strong lensing.

`lenstronomy` includes, but is not limited to, the methods presented in (Birrer et al., 2015). This includes a linear source reconstruction method based on Shapelet (Refregier, 2003) basis sets, a Particle Swarm Optimization (Eberhart and Kennedy, 1995) for optimizing the non-linear lens model parameters and a MCMC framework for Bayesian parameter inference (emcee Foreman-Mackey et al., 2013). The software supports a high dynamic range in angular scales, complexity in source and lens models, can handle various image qualities and meets the requirements for diverse science applications. Furthermore, `lenstronomy` enables a consistent integration of imaging, time-delay and kinematic data to provide model constraints.

There is continued development and support of `lenstronomy` due to the ongoing work of multiple active users. This includes modelling a doubly lensed quasar (Birrer et al. in prep) for a cosmographic inference, modelling a set of quadruply lensed quasars in multi-band images (Shajib et al. in prep), modelling quasar flux ratios in multi-plane lensing (Gilman et al. in prep), modelling wide field survey lens candidates based on ground based imaging

(Treu et al. submitted) and in cluster lens modelling of a giant arc in a high redshift cluster (Birrer et al. in prep). The performance of the source reconstruction capabilities has been compared with other publicly available software (Joseph et al. submitted) and was found to behave well in speed and reconstruction accuracy.

This paper is structured as follows: In Section 2, we provide an overview of the software architecture and deployment, including installation and dependencies. In Section 3, we describe the core modules of `lenstronomy` with some simple examples of how to use them. We provide some modelling examples in Section 4 to demonstrate the capabilities and flexibility of `lenstronomy`. In Section 5, we give some science application highlights. We summarize in Section 6.

This publication is accompanied by a public release of the source code with extensive online-documentation¹. Additionally, we provide Jupyter² notebook examples for different science cases. We refer the reader to these online resources for the most updated version of the software package.

2. Package overview

`lenstronomy` is an open source software package developed in python (Rossum, 1995). Distribution is granted through the MIT open source software licence. `lenstronomy` relies on packages included in the python standard library³ and the proven open-source libraries `numpy`⁴, `scipy`⁵, `astropy` (Astropy Collaboration et al., 2013), and `matplotlib` (Hunter, 2007). For optimization and parameter inference routines, `CosmoHammer` (Akeret et al., 2013) is used, which supports Message Passing Interface (MPI) and allows for massive parallelization of the `emcee` (Foreman-Mackey et al., 2013) algorithm.

The stable release version can be installed at the command line via `pip`:

```
$ pip install lenstronomy --user
```

¹<https://lenstronomy.readthedocs.io>

²<http://jupyter.org>

³<https://docs.python.org/2/library/index.html>

⁴www.numpy.org

⁵www.scipy.org

lenstronomy is compatible with both python2⁶ and python3. The code design and development follow effective practices for scientific computing. The development is coordinated on GitHub⁷ and stable versions are released through PyPi⁸ - the python packaging index. This full integration of lenstronomy in the python packaging environment allows third party to release and share software packages, analysis routines and work-flows that rely upon lenstronomy.

Extensive tests have been added to lenstronomy which allows us to develop the package using a Continuous Integration (CI) process. The tests are performed on a virtual environment⁹ to ensure cross-platform stability and the test coverage report is publicly available¹⁰. Documentation for the various routines and classes are provided through sphinx¹¹ and are available on ReadtheDocs¹². To make use of lenstronomy, a start guide and several application examples are provided in the form of Jupyter notebooks in an extension module¹³.

lenstronomy is structured in multiple independent modules, each consisting of multiple classes and sub-packages. These modules execute specific tasks and come with well-defined interfaces. Among the core modules are:

1. `LensModel`: Provides the lensing functionalities. The full functionality is supported with an arbitrary superposition of individual lens models (Section 3.1).
2. `LightModel`: Enables a variety of surface brightness descriptions and profiles. See Section 3.2.
3. `PointSource`: Handles the point sources (Section 3.3).
4. `Data`: Handling all data specific tasks. Including Point-Spread function (PSF), coordinate systems and noise properties (Section 3.4).
5. `ImSim`: Simulates images. Queries the specifications made in `LensModel`, `LightModel`, `PointSource` and `Data` (Section 3.5).

⁶python2 will not be maintained past 2020

⁷<https://github.com/sibirrer/lenstronomy>

⁸<https://pypi.python.org/pypi/lenstronomy>

⁹<https://travis-ci.org/sibirrer/lenstronomy>

¹⁰<https://coveralls.io/github/sibirrer/lenstronomy>

¹¹www.sphinx-doc.org

¹²<https://lenstronomy.readthedocs.io>

¹³https://github.com/sibirrer/lenstronomy_extensions

6. `Workflow`: Higher level interface to define fitting sequences and infer model parameters based on simulations of `ImSim` (Section 3.6).
7. `GalKin`: Computes (stellar) kinematics of the deflector galaxy with spherical Jeans modeling based on the mass model specified in `LensModel` and the lens light model specified in `LightModel` (Section 3.7).

The core modules perform the individual tasks associated with lens modeling. Each module can be used as a stand-alone package and various extension modules are available. The strength of `lenstronomy` is the full integrated support of each individual module when it comes to lens modeling.

In Section 3, we briefly describe the main functionalities of the core modules and provide some simple use cases. The interplay between the modules is demonstrated in Section 4 and 5.

3. Core modules of `lenstronomy`

In the following, we describe the basic functionalities of the most important modules of `lenstronomy` with some simple examples. More detailed information about the available routines and their use can be accessed through the online documentation.

3.1. *LensModel module*

`LensModel` and its sub-packages execute all the purely lensing related tasks of `lenstronomy`. This includes ray-shooting, solving the lens equation, arrival time computation and non-linear solvers to optimize lens models for specific image configurations. The module allows consistent integration with single and multi plane lensing and an arbitrary superpositions of lens models. There is a wide range of lens models available. For details we refer the reader to the online-documentation.

To demonstrate the design of `LensModel`, we initialize a lens model and then execute some lensing calculations. First, we perform these calculations in a single-plane configuration 3.1.1 and then in a multi-plane configuration 3.1.2. Then we demonstrate the lens equation solver, that can be applied in both cases with the same interface 3.1.3.

3.1.1. Single-plane lensing

The default setting of `LensModel` is to operate in single lens plane mode, where the superpositions of multiple lens models are de-coupled. Below we provide an example of a lens model, that consists of a super-position of an elliptical power-law potential, an external shear and an additional singular isothermal sphere perturber. We initialize the `LensModel` class, define the parameters for each individual model and perform some standard lensing calculations, such as a backwards ray-shooting of an image plane coordinate, computation of the Fermat potential and evaluating the magnification.

```
1 # import the LensModel class #
2 from lenstronomy.LensModel.lens_model import LensModel
3
4 # specify the choice of lens models #
5 lens_model_list = ['SPEMD', 'SHEAR', 'SIS']
6
7 # setup lens model class with the list of lens models #
8 lensModel = LensModel(lens_model_list=lens_model_list)
9
10 # define parameter values of lens models #
11 kwargs_spep = {'theta_E': 1., 'e1': 0.06, 'e2': 0., '
    ↪ gamma': 2., 'center_x': 0.1, 'center_y': 0}
12 kwargs_shear = {'e1': 0.02, 'e2': 0.0}
13 kwargs_sis = {'theta_E': 0.2, 'center_x': 1., 'center_y':
    ↪ -0.1}
14 kwargs_lens = [kwargs_spep, kwargs_shear, kwargs_sis]
15
16 # image plane coordinate #
17 theta_ra, theta_dec = 1.4, .3
18
19 # source plane coordinate #
20 beta_ra, beta_dec = lensModel.ray_shooting(theta_ra,
    ↪ theta_dec, kwargs_lens)
21 # Fermat potential #
22 fermat_pot = lensModel.fermat_potential(x_image=theta_ra,
    ↪ y_image=theta_dec, x_source=beta_ra, y_source=
    ↪ beta_dec, kwargs_lens=kwargs_lens)
23
24 # Magnification #
```

```

25 | mag = lensModel.magnification(theta_ra, theta_dec,
    |     ↪ kwargs_lens)

```

Additionally, the `LensModel` class allows to compute the Hessian matrix, shear and convergence, deflection angle and lensing potential. These routines are fully compatible with the numpy array structure and superposition of an arbitrary number of lens models.

3.1.2. Multi-plane lensing

The multi-plane setting of `LensModel` allows the user to place several deflectors at different redshifts. When not further specified, the default cosmology used is that of the `astropy` cosmology class. The interface to access the lensing functionalities remains the same as for the single-plane setting 3.1.1. As an example, we take the same setting as in 3.1.1 but place the singular isothermal sphere perturber at a lower redshift.

```

1 | # keep the imports and variables from above #
2 | # specify redshifts of deflectors #
3 | redshift_list = [0.5, 0.5, .1]
4 | # specify source redshift #
5 | z_source = 1.5
6 | # setup lens model class with the list of lens models #
7 | lensModel_mp = LensModel(lens_model_list=lens_model_list,
    |     ↪ z_source=z_source, redshift_list=redshift_list,
    |     ↪ multi_plane=True)
8 |
9 | # source plane coordinate #
10 | beta_ra, beta_dec = lensModel_mp.ray_shooting(theta_ra,
    |     ↪ theta_dec, kwargs_lens)
11 |
12 | # Magnification #
13 | mag = lensModel_mp.magnification(theta_ra, theta_dec,
    |     ↪ kwargs_lens)

```

3.1.3. Lens equation solver

Solving the lens equation to compute the (multiple) image positions of a given source position can be conveniently performed within `LensModel` and is supported with a general instance of the `LensModel` class.

```

1 | # keep the imports and variables from above #
2 | # import the lens equation solver class #

```

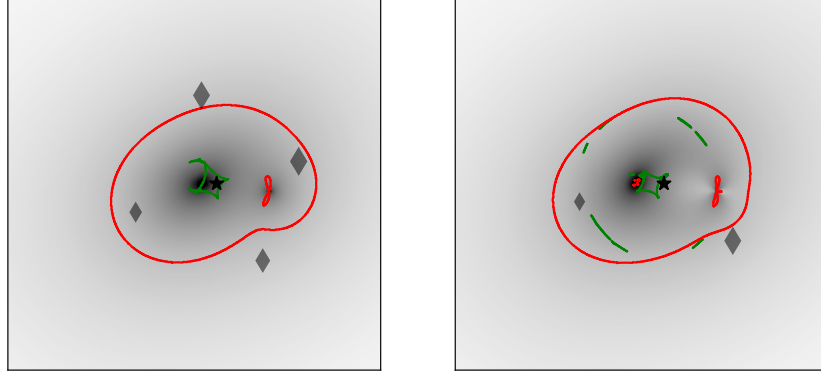



Figure 1: Illustrations of the two lens models created in Section 3.1. **Left:** The single-plane lens model of 3.1.1. **Right:** The multi-plane lens model of 3.1.2. In gray-scale the convergence map of the lens models, red lines correspond to the critical curve, green lines to the caustics. The green star corresponds to the source positions and the diamonds to the corresponding image positions.

```

3 from lenstronomy.LensModel.Solver.lens_equation_solver
   ↪ import LensEquationSolver
4
5 # specifiy the lens model class to deal with #
6 solver = LensEquationSolver(lensModel)
7
8 # solve for image positions provided a lens model and the
   ↪ source position #
9 theta_ra, theta_dec = solver.image_position_from_source(
   ↪ beta_ra, beta_dec, kwargs_lens)
10
11 # the magnification of the point source images #
12 mag = lensModel.magnification(theta_ra, theta_dec,
   ↪ kwargs_lens)

```

Two lens models are shown in Figure 1. The source position of the example and the solutions of the lens equation (image positions) are marked.

3.2. LightModel module

The `LightModel` class provides the functionality to describe galaxy surface brightnesses. `LightModel` supports various analytic profiles as well as representations in shapelet basis sets. Any superposition of different profiles is supported. We refer to the online documentation for the full list of surface brightness profiles available and their parameterisation.

As an example, we initialize two `LightModel` class, one with a spherical Sersic profile and one with an elliptical Sersic profile. We define the profile parameters and evaluate the surface brightness at a specific position. The two `LightModel` instances will later be used as the lens light and the source light.

```
1 # import the LightModel class #
2 from lenstronomy.LightModel.light_model import LightModel
3 # set up the list of light models to be used #
4 source_light_model_list = ['SERSIC']
5 lightModel_source = LightModel(light_model_list=
    ↪ source_light_model_list)
6 lens_light_model_list = ['SERSIC_ELLIPSE']
7 lightModel_lens = LightModel(light_model_list=
    ↪ lens_light_model_list)
8 # define the parameters #
9 kwargs_light_source = [{'I0_sersic': 10, 'R_sersic':
    ↪ 0.02, 'n_sersic': 1.5, 'center_x': beta_ra, '
    ↪ center_y': beta_dec}]
10 # ellipticity parameter are defined as eccentricities #
11 import lenstronomy.Util.param_util as param_util
12 e1, e2 = param_util.phi_q2_ellipticity(phi=0.5, q=0.7)
13 kwargs_light_lens = [{'I0_sersic': 1000, 'R_sersic': 0.1,
    ↪ 'n_sersic': 2.5, 'e1': e1, 'e2': e2, 'center_x':
    ↪ 0.1, 'center_y': 0}]
14
15 # evaluate surface brightness at a specific position #
16 flux = lightModel_lens.surface_brightness(x=1, y=1,
    ↪ kwargs_list=kwargs_light_lens)
```

3.3. PointSource module

To accurately predict and model the positions and fluxes of point sources, different numerical procedures are needed compared to extended surface

brightness features. The `PointSource` module manages the different options in describing point sources (e.g. in the image plane or source plane, with fixed magnification or allowed with individual variations thereof) and provides a homogeneous interface to access image positions and magnifications. The `PointSource` class requires an instance of a `LensModel` class in case of lensed sources and arbitrary superpositions of point sources are allowed.

In the example below, we create two instances of the `PointSource` class. One with a parameterization in the source plane and one with a parameterization in the image plane. The interface to access the necessary information about the image positions and magnifications remain the same in both cases.

```

1 # import the PointSource class #
2 from lenstronomy.PointSource.point_source import
   ↪ PointSource
3
4 # unlensed source position #
5 point_source_model_list = ['SOURCE_POSITION']
6 pointSource = PointSource(point_source_type_list=
   ↪ point_source_model_list, lensModel=lensModel,
   ↪ fixed_magnification_list=[True])
7 kwargs_ps = [{'ra_source': beta_ra, 'dec_source':
   ↪ beta_dec, 'source_amp': 100}]
8 # return image positions and amplitudes #
9 x_pos, y_pos = pointSource.image_position(kwargs_ps=
   ↪ kwargs_ps, kwargs_lens=kwargs_lens)
10 point_amp = pointSource.image_amplitude(kwargs_ps=
   ↪ kwargs_ps, kwargs_lens=kwargs_lens)
11
12 # lensed image positions (solution of the lens equation)
   ↪ #
13 point_source_model_list = ['LENSED_POSITION']
14 pointSource = PointSource(point_source_type_list=
   ↪ point_source_model_list, lensModel=lensModel,
   ↪ fixed_magnification_list=[False])
15 kwargs_ps = [{'ra_image': theta_ra, 'dec_image':
   ↪ theta_dec, 'point_amp': np.abs(mag)*30}]
16 # return image positions and amplitudes #
17 x_pos, y_pos = pointSource.image_position(kwargs_ps=
   ↪ kwargs_ps, kwargs_lens=kwargs_lens)

```

```

18 point_amp = pointSource.image_amplitude(kwargs_ps=
    ↪ kwargs_ps, kwargs_lens=kwargs_lens)

```

3.4. Data module

The Data module consists of two main classes. The Data class stores and manages all the imaging data relevant information. This includes the coordinate frame, coordinate-to-pixel transformation (and the inverse), and, in the case of fitting, also noise properties for computing the likelihood of the data given the model. The PSF class handles the point spread function convolution. Supported are pixelised convolution kernels as well as some analytic profiles.

```

1 # import the Data() class #
2 from lenstronomy.Data.imaging_data import Data
3 deltaPix = 0.05 # size of pixel in angular coordinates #
4
5 # setup the keyword arguments to create the Data() class
    ↪ #
6 kwargs_data = {'numPix': 100,
7               'ra_at_xy_0': -2.5,
8               'dec_at_xy_0': -2.5,
9               'transform_pix2angle': np.array([[1, 0], [0, 1]]) *
    ↪ deltaPix}
10 data = Data(kwargs_data)
11 # return the list of pixel coordinates #
12 x_coords, y_coords = data.coordinates
13 # compute pixel value of a coordinate position #
14 x_pos, y_pos = data.map_coord2pix(ra=0, dec=0)
15 # compute the coordinate value of a pixel position #
16 ra_pos, dec_pos = data.map_pix2coord(x=20, y=10)
17
18 # import the PSF() class #
19 from lenstronomy.Data.psf import PSF
20 kwargs_psf = {'psf_type': 'GAUSSIAN', 'fwhm': 0.1, '
    ↪ pixel_size': deltaPix}
21 psf = PSF(kwargs_psf)
22 # return the pixel kernel corresponding to a point source
    ↪ #
23 kernel = psf.kernel_point_source

```

3.5. *ImSim module*

At the core of the `ImSim` module is the `ImageModel` class. `ImageModel` is the interface to combine all the different components, `LensModel`, `LightModel`, `PointSource` and `Data` to model images. The `LightModel` can be used to model both lens light (un-lensed) and source light (lensed) components. `ImSim` supports all functionalities of each of those components. `ImageModel` is supported by the class `ImageNumerics` that specifies and executes the numerical options accessible. Among the numerical options are sub-pixel grid resolution ray-tracing and convolutions that can improve numerical accuracy in the presence of either small lensing perturbations and/or a highly variable surface brightness profile (see e.g. Tessore et al., 2016, for the latter).

3.5.1. *Image simulation*

As an example, we simulate an image with an instance of `ImageModel` that use instances of the classes we created above. We can define two different `LightModel` instances for the lens and source light. We define the sub-pixel ray-tracing resolution and whether the PSF convolution is applied on the higher resolution ray-tracing grid or on the degraded pixel image.

```
1 # import the ImageModel class #
2 from lenstronomy.ImSim.image_model import ImageModel
3 # define the numerics #
4 kwargs_numerics = {'subgrid_res': 3,
5                    'psf_subgrid': True}
6 # initialize the Image model class by combining the
7   ↪ modules we created above #
8 imageModel = ImageModel(data_class=data, psf_class=psf,
9   ↪ lens_model_class=lensModel,
10   ↪ source_model_class=
11   ↪ lightModel_source,
12   ↪ lens_light_model_class=
13   ↪ lightModel_lens,
14   ↪ point_source_class=pointSource,
15   ↪ kwargs_numerics=kwargs_numerics)
16 # simulate image with the parameters we have defined
17   ↪ above #
18 image = imageModel.image(kwargs_lens=kwargs_lens,
19   ↪ kwargs_source=kwargs_light_source,
```

```

14         kwargs_lens_light=
           ↪ kwargs_light_lens,
           ↪ kwargs_ps=kwargs_ps)
15
16 # we can also add noise #
17 import lenstronomy.Util.image_util as image_util
18 # exposure time to quantify the Poisson noise level #
19 exp_time = 100
20 # background rms value #
21 background_rms = 0.1
22 poisson = image_util.add_poisson(image, exp_time=exp_time
           ↪ )
23 bkg = image_util.add_background(image, sigma_bkd=
           ↪ background_rms)
24 image_noisy = image + bkg + poisson

```

Figure 2 shows the simulated image of the example computed above with the single-plane lens model of Section 3.1.1 (left panel) and for the same light profiles but with the multi-plane lens model of Section 3.1.2 in the right panel. To illustrate the numerical procedure in how `lenstronomy` renders images, we provide another example consisting of a high resolution galaxy profile in Figure 3.

3.5.2. Linear inversion

Parameters corresponding to an amplitude of a surface brightness distribution have a linear response on the predicted flux values of pixels and can be inferred by a linear minimization based on the data. `lenstronomy` automatically identifies those parameters. The `ImSim` module comes with an option such that the linear parameters do not have to be provided when fitting a model to data. This can reduce the number of non-linear parameters significantly, depending on the source complexity to be modelled. In the example provided in Section 3.5.1, we have 6 linear parameters, the 4 point source amplitudes and the amplitudes of the Sersic profile of the lens and source. To perform the linear inversion, noise properties of the data have to be known or assumed 3.5.3. There are different approaches in the literature that perform different types of semi-linear inversions (e.g. Suyu et al., 2006; Vegetti and Koopmans, 2009; Tagore and Keeton, 2014; Birrer et al., 2015; Nightingale and Dye, 2015).

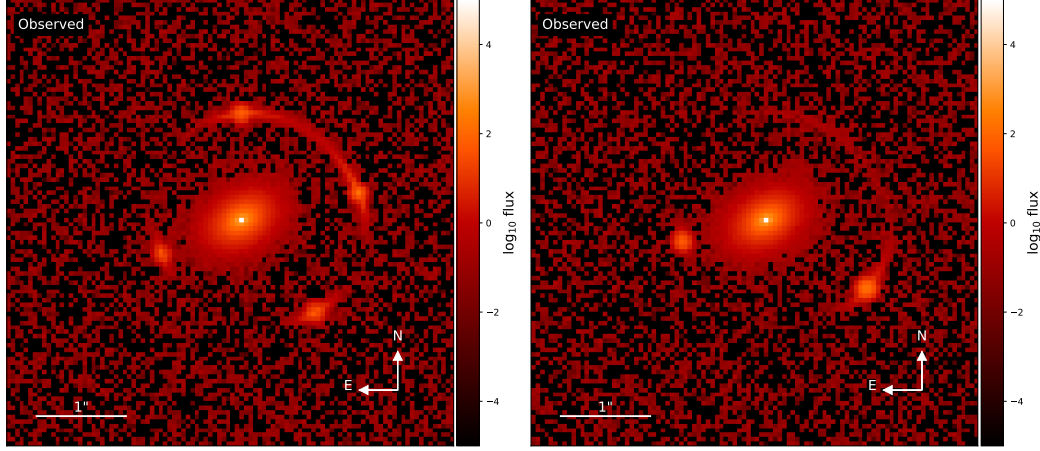


Figure 2: Simulated images with the chosen options of the `ImageModel` class (3.5). **Left:** The single-plane lens model results in a quadruply imaged quasar. **Right:** The multi-plane model results in a double lensed quasar.

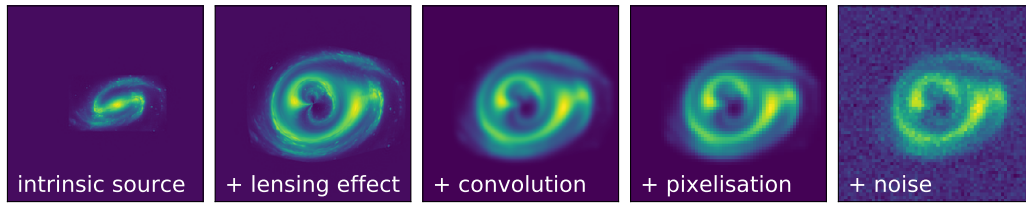


Figure 3: Illustration of the sequence of steps performed by the `ImSim` module. **From left to right:** A galaxy described with a high resolution shapelet representation (1), seen through gravitational lensing, i.e. evaluated in the image plane coordinates (2), convolved on a subpixel high resolution grid (3), down-graded to pixel resolution and added with noise (5).

3.5.3. Likelihood

The likelihood of the data given a model $p(d_{\text{data}}|d_{\text{model}})$ is key in sampling the parameter posterior distribution (see Section 3.6) and also to perform the linear inversion (Section 3.5.2). The convention `lenstronomy` uses to compute $p(d_{\text{data}}|d_{\text{model}})$ is

$$\log p(d_{\text{data}}|d_{\text{model}}) = \sum_i \frac{(d_{\text{data},i} - d_{\text{model},i})^2}{2\sigma_i^2} + \text{const.} \quad (1)$$

The constant term in equation 1 is not computed by `lenstronomy`. The error in each pixel, σ_i , consists of a Gaussian background term, σ_{bkgd} , and a Poisson term based on the count statistics of an individual pixel, f_i , such that $d_{\text{model},i}/f_i$ is the Poisson error predicted by the model in the time units of the data, and writes

$$\sigma_i^2 = \sigma_{\text{bkgd}}^2 + d_{\text{model},i}/f_i. \quad (2)$$

In our example of 3.5.1, f_i is the exposure time for each pixel and σ_{bkgd} is the background rms value. CCD gain and other components may be incorporated into f_i .

The linear inversion requires an estimate of the noise term, σ_i^2 , without the knowledge of the model, $d_{\text{model},i}$. For this particular step, the linear inversion is performed based on the Poisson noise expected by the data itself

$$\sigma_{\text{linear}, i}^2 = \sigma_{\text{bkgd}}^2 + d_{\text{data},i}/f_i. \quad (3)$$

The analytic marginalization over the covariance matrix of the linear inversion (Gaussian approximation) can be added (see Birrer et al., 2015, for further information). Additionally, pixel masks can be set and additional error terms can be plugged in, if required.

3.6. Workflow module

The `Workflow` module manages the execution of the non-linear fitter (in our case the PSO) and the parameter inference (in our case the `CosmoHammer` `emcee`). The two main classes are `Param` and `FittingSequence`. These classes handle all the model choices of the user and migrate them to the external modules and from the external modules back to `lenstronomy`.

3.6.1. *parameter handling*

The `Param` class is the interface of the `lenstronomy` conventions of parameters (list of keyword arguments) and the fitting conventions of `CosmoHammer` or `emcee` (single array consisting all parameters). The `Param` class enables the user to set further options:

1. keep certain parameters fixed
2. handling of the linear parameters
3. provide additional constraints on the modelling (e.g. fix source profile to point source position etc)

With the `Param` class, `lenstronomy` can be integrated robustly also with other sampling methods.

3.6.2. *Parameter fitting*

The `FittingSequence` class allows the user to perform a PSO and/or MCMC run. The user can run a sequence of fitting routines, applied one after the other with taking the results of the previous routine as an input of the next one. The user can specify (optionally) to keep one or multiple parameter classes (lens model, source model, lens light model and source model) fixed during the fitting process or parameter inference.

The user further needs to specify:

1. define input guess and uncertainties to start sampling the parameter space
2. provide hard bounds on parameters not to be considered (as upper and lower limits)
3. set the number of particles and number of iterations being performed by the PSO/MCMC

`FittingSequence` does not rely on instances of other `lenstronomy` classes. It can be constructed from elements in the form of keyword arguments and lists thereof in the `lenstronomy` conventions. This enables a safe and reliable execution of tasks within `FittingSequence` on non-local platforms, such as high performance clusters.

3.7. *GalKin module*

Kinematics of the lensing galaxy can provide additional constraints on the lens model and can help to reduce systematics inherent in lensing. The `GalKin` module provides the support to self-consistently model and predict

the velocity dispersion of the lensing galaxy given the surface brightness profile and the lens model upon which the image modelling consists of. The kinematics require the knowledge/assumption of the 3d light and mass profiles. Not all lens and light models can be analytically de-projected. In these cases, `lenstronomy` performs a Multi-Gaussian decomposition (Cappellari, 2002) and the de-projection is performed on the individual Gaussian components. The kinematics is computed with spherical Jeans anisotropy modelling (JAM). `lenstronomy` supports the stellar anisotropy profiles described in Mamon and Lokas (2005). Observational conditions, i.e. the PSF and the aperture are modelled with a spectral rendering approach described in Birrer et al. (2017a).

4. Modelling examples

The design of `lenstronomy` and the core modules described in Section 3 allow a wide range of modelling tasks to be executed, that go beyond the classical lens modelling. In this section, we provide five examples in different domains where we demonstrate the capabilities of `lenstronomy`.

4.1. Source reconstruction

Reconstruction techniques are required to describe the source morphology at the scales relevant for given data. In other words, if the source reconstruction representation is insufficient in describing the data even in the limit of the correct lens model, the lens model inference can be significantly biased. In Figure 4, we provide an example where we reconstruct a source galaxy with complex morphology with a Shapelet basis set. We are able to represent the features present in the image. The reconstruction of the source reproduces the macroscopic morphology of the input galaxy.

4.2. Image de-convolution

The source reconstruction in Section 4.1 is a combination of two distinct steps: a de-lensing (effectively a non-linear mapping between the image plane and the basis set represented in the source plane) (1) and a de-convolution (2). By removing the class instance of `LensModel` from the `ImSim` module, the linear inversion method built in `lenstronomy` effectively performs a de-convolution. This is demonstrated in Figure 5 where we take a scaled version of the same galaxy as for Figure 4 with a scaled PSF convolution kernel and apply the same shapelet basis set to describe the image. The deconvolved

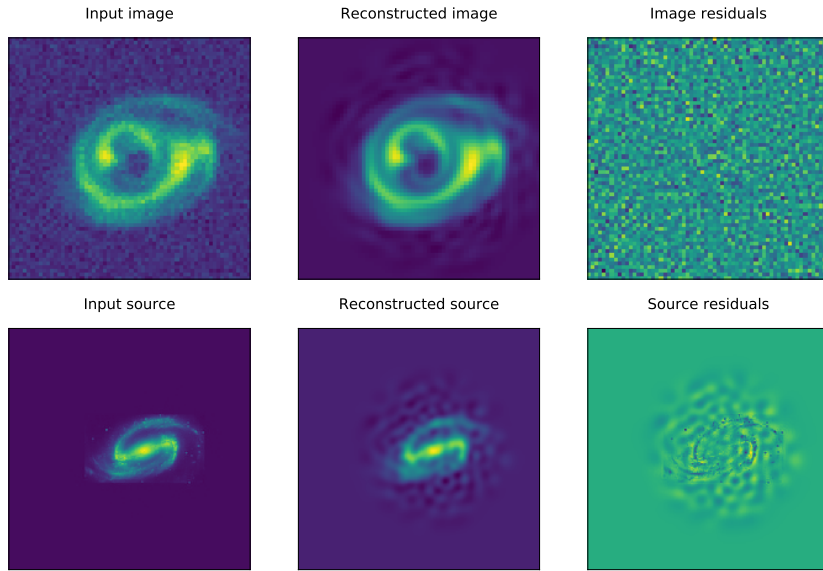


Figure 4: lenstronomy source reconstruction capabilities (de-lensing + de-convolution). **Top:** Image plane. **bottom:** Source plane. **From left to right:** The simulated image with noise based on a realistic galaxy surface brightness profile (same example as Figure 3) (1), the reconstruction by the linear inversion technique based on a lower resolution shapelet basis set (2), the residuals of the input relative to the reconstructed model. The lens model was kept fixed to the input model for the reconstruction.

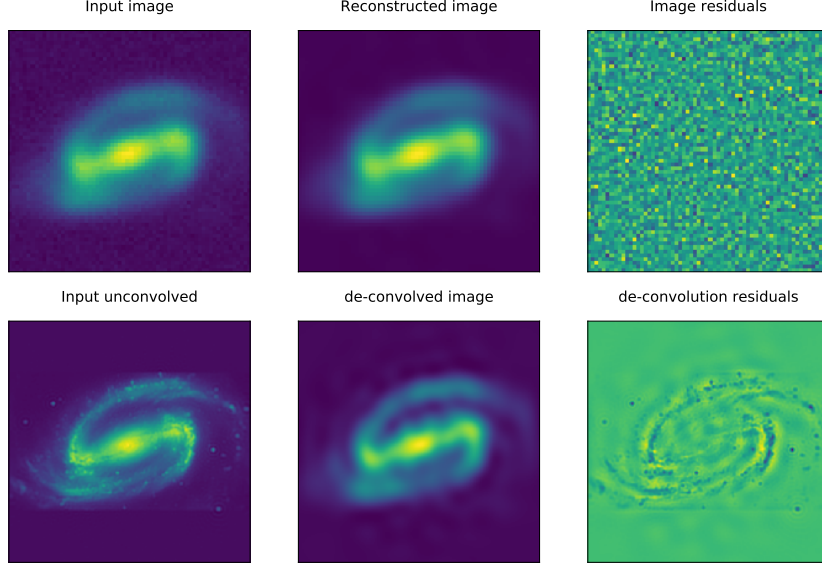


Figure 5: `lenstronomy` de-convolution capabilities (source reconstruction without lensing). **Top:** Convolved images. **bottom:** Unconvolved images. **From left to right:** The simulated image with noise based on a realistic galaxy surface brightness profile (same example as Figure 3 but scaled and with a larger convolution kernel applied) (1), the reconstruction by the linear inversion technique based on a lower resolution shapelet basis set (2), the residuals of the input relative to the reconstructed model.

reconstruction is sharpened. Deconvolution only does not benefit from the lensing magnification.

4.3. Galaxy structural analysis

`lenstronomy` can be used to extract structural components from galaxy images. This is yet another example where the lensing capabilities of `lenstronomy` do not have to be used necessarily. In terms of flexibility, `lenstronomy` contains similar features and flexibilities as the well established software `GALFIT` (Peng et al., 2002, 2010). `lenstronomy` provides an open source alternative in `python`. We also emphasize that `lenstronomy` comes along with an MCMC algorithm that can provide covariances between inferred parameters. Additionally, `lenstronomy` is able to extract structural parameters from

lensed and highly distorted galaxies.

4.4. Quasar-host galaxy decomposition

In the case where the galaxy contains a quasar, simultaneous decompositions of the host galaxy and a point source component can be performed with `lenstronomy`. Figure 6 demonstrates this capability. A joint fitting of two component Sersic profile for the host galaxy and a quasar point source were used as an input model and the different components were recovered in the modelling.

4.5. Multiband fitting

`lenstronomy` is explicitly designed to simultaneously model lenses in multiple bands. The coordinate system definition is image independent and can be shared among multiple data sets. The `FittingSequence` class (see 3.6.2) naturally handles an arbitrary number of data sets, all with their own descriptions (see 3.4). The non-linear parameters, such as lens model, point source position and light profile shapes are shared among the different bands. The linear parameters, however, are optimized for each band individually. This allows e.g. for different galaxy morphologies in different wavelength. The multi-band approach allows also to model a set of single exposures directly rather than rely on combined post-processed data products. This approach can also be used to model disjoint patches of a cluster arc without requiring a large image.

Precise relative astrometry may be required to perform the lens modelling in a joint coordinate frame. `lenstronomy` comes with an iterative routine to align coordinate frames from different bands given a shared model description. This can be used to align images to determine e.g. a point source or the lensing galaxy light center.

5. Science applications of `lenstronomy`

In this section we provide science examples that `lenstronomy` has enabled. In particular, we will highlight specific settings within `lenstronomy` that were required to conduct the analysis in the domain of substructure lensing, time-delay cosmography and non-linear shear measurements.

As the size and diversity of known strong lensing systems increases, a wider variety of science topics can be tackled, such as time-delay cosmography with lensed SNIa Grillo et al. (2018), single star micro-lensing cluster arcs

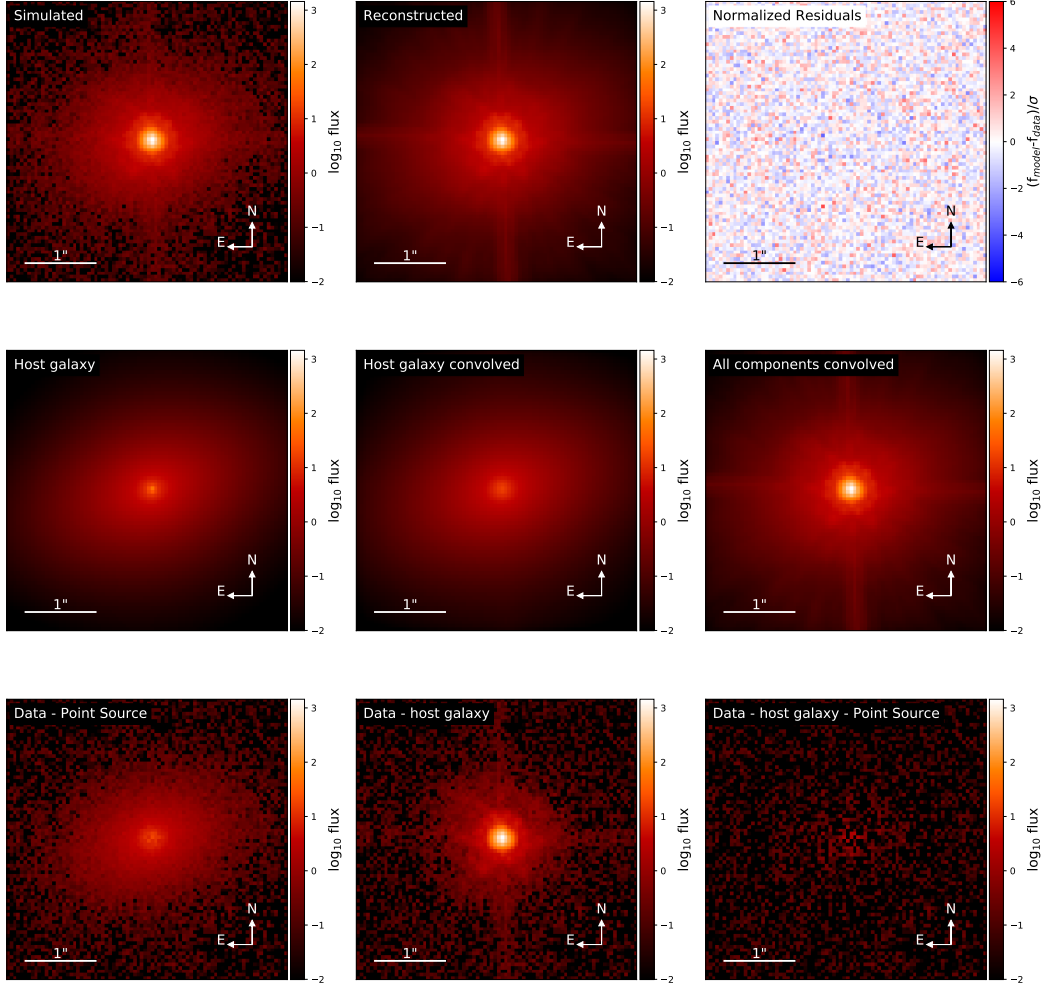


Figure 6: Illustration of the `lenstronomy` quasar-host galaxy decomposition capabilities. A two-component host galaxy is modelled with a significantly brighter quasar at its center. `lenstronomy` can reliably decompose the two components. **Top:** Simulated image (left), reconstructed model (middle) and residuals of the reconstruction (right). **Middle:** The separation of the model components. **Bottom:** The subtraction of the model components from the data.

Kelly et al. (2017), double source-plane cosmography Gavazzi et al. (2008); Collett and Auger (2014) or cosmic shear measurements with Einstein rings (Birrer et al., 2018).

We emphasize that the choices when modelling a specific system remains the task of the user. `lenstronomy` may facilitate scientific analysis of strong lenses, but should be accompanied by rigorous testing of the specific method applied, desirably through simulations. `lenstronomy` allows to simulate accurate mock data with very complex structure in lens and source and therefore facilitate the exploration of systematics in the analysis.

5.1. Lensing substructure quantification

Substructure is expected to perturb the deflection angle below the pixel resolution of an HST image. To detect and/or quantify those astrometric anomalies, the numerical accuracy has to keep up with this small effect. Sub-grid resolution ray-tracing is required to perform such analysis on HST images. Additionally, the source surface brightness resolution captured by the model must be sufficiently high resolution not to impact the analysis. This is especially true in regions where a proposed perturber may affect image fluxes. In Birrer et al. (2017a), we specifically enhanced the source reconstruction resolution where we proposed a clump to be present.

5.2. Time-delay cosmography

The workflow interface facilitates a fast exploration of various choices and options in all the aspects of lens modelling. It is necessary to explore the degeneracies inherent in lensing and their impact on the cosmographic inference. In Birrer et al. (2016), we explored the source scale degeneracy by explicitly mapping out the source size with the shapelet scale parameter. The built-in time-delay likelihood and the `GALKin` module provide the full support for a fully self-consistent analysis of imaging, time-delay and kinematic data to derive cosmographic constraints.

5.3. Cosmic shear measurements

We applied `lenstronomy` to model and reconstruct the non-linear shear distortions that couple to the main deflector in an Einstein ring lens in the COSMOS field (Birrer et al., 2017b). The detailed modelling of the HST imaging of the Einstein ring allowed us to constrain the shear parameters to very high precision. `lenstronomy` is aimed to have the flexibility to model hundreds or even the aimed thousands of Einstein ring lenses expected in

future space based surveys to provide comparable and complementary cosmic shear measurements, as been fore-casted by Birrer et al. (2018).

6. Conclusion

We have presented `lenstronomy`, a multi-purpose open source lens modelling software package in python. We outlined its design and the major supported features. `lenstronomy` has been used to study the expansion history of the universe with time-delay cosmography and to probe dark matter properties by substructure lensing. The modular nature of `lenstronomy` provides support for a wide range of scientific studies. We have provided modelling and science examples to illustrate some of the capabilities of `lenstronomy`. The software is distributed under the MIT license. The software is actively used and maintained and the latest stable release will be distributed through the python packaging index. We refer to the online documentation¹⁴, where the latest starter guide, example notebooks, source code and installation guidelines can be found.

Acknowledgements

SB thanks Alexandre Refregier and Tommaso Treu for useful comments and support that enabled the development and public release of `lenstronomy`. SB thanks Joel Akeret for valuable software engineering support and advices in design and good practice software development. The software was and is actively in use by multiple users. We especially thank Anowar Shajib, Daniel Gilman, Felix A. Kuhn, Kevin Fusshoeller, Cyril Welschen, Felix Mayor, Remy Joseph, Martin Millon, Xuheng Ding and Brian Nord. Their valuable feedback lead to improvements in stability, bug fixing, more general design implementations and extended the capabilities of `lenstronomy`.

SB acknowledges support from NASA through grant HST-GO-14254 and HST-GO-14630 from the Space Telescope Science Institute, which is operated by the Association of Universities for Research in Astronomy, Inc., under NASA contract NAS 5-26555.

¹⁴<https://lenstronomy.readthedocs.io>

Appendix A. Publicly available lens modelling software

A collection of public available lens modelling software presented in the literature is listed below. We refer to specific literature and online documentations for the scope of each individual software and its current development status.

- `gravlens` (Keeton, 2011)¹⁵: A standard lens model software widely used in the community. Includes a wide range of basic lensing calculations and comes with an extension that adds many routines for modeling strong lenses.
- `lenstool` (Kneib et al., 2011)¹⁶: A g Lensing software for modeling mass distribution of galaxies and clusters. Comes with a Bayesian inference method.
- `PixeLens` (Saha and Williams, 2011)¹⁷: A program for reconstructing gravitational lenses from multiple-imaged point sources. It can explore ensembles of lens-models consistent with given data on several different lens systems at once.
- `glafic` (Oguri, 2010)¹⁸: Support for many mass models and parametric light models. Simulates lensed extended images with PSF convolution.
- `LENSED` (Tessore et al., 2016)¹⁹: Performs forward parametric modelling of strong lenses. Supports computing on GPUs.
- `AutoLens` (Nightingale et al., 2017)²⁰: An automated modeling suite for the analysis of galaxy-scale strong gravitational lenses. Incorporates an adaptive grid source reconstruction technique.
- `Ensai` (Hezaveh et al., 2017)²¹: Estimating parameters of strong gravitational lenses with convolutional neural networks.

¹⁵<http://www.physics.rutgers.edu/~keeton/gravlens/>

¹⁶<http://projets.lam.fr/projects/lenstool/wiki>

¹⁷<http://www.physik.uzh.ch/~psaha/lens/pixelens.php>

¹⁸<http://www.slac.stanford.edu/~oguri/glafic/>

¹⁹<http://glenco.github.io/lensed/>

²⁰<http://jamesnightingale.net/AutoLens/>

²¹<https://github.com/yasharhezaveh/Ensai>

- pySPT (Wertz and Orthen, 2018)²²: A package dedicated to the Source Position Transformation (SPT). The main goal of pySPT is to provide a tool to quantify the systematic errors that are introduced by the SPT in lens modeling.

References

- Agnello, A., Treu, T., Ostrovski, F., Schechter, P. L., Buckley-Geer, E. J., Lin, H., Auger, M. W., Courbin, F., Fassnacht, C. D., Frieman, J., Kuropatkin, N., Marshall, P. J., McMahon, R. G., Meylan, G., More, A., Suyu, S. H., Rusu, C. E., Finley, D., Abbott, T., Abdalla, F. B., Allam, S., Annis, J., Banerji, M., Benoit-Lévy, A., Bertin, E., Brooks, D., Burke, D. L., Carnero Rosell, A., Carrasco Kind, M., Carretero, J., Cunha, C. E., D’Andrea, C. B., da Costa, L. N., Desai, S., Diehl, H. T., Dietrich, J. P., Doel, P., Eifler, T. F., Estrada, J., Fausti Neto, A., Flaugh, B., Fosalba, P., Gerdes, D. W., Gruen, D., Gutierrez, G., Honscheid, K., James, D. J., Kuehn, K., Lahav, O., Lima, M., Maia, M. A. G., March, M., Marshall, J. L., Martini, P., Melchior, P., Miller, C. J., Miquel, R., Nichol, R. C., Ogando, R., Plazas, A. A., Reil, K., Romer, A. K., Roodman, A., Sako, M., Sanchez, E., Santiago, B., Scarpine, V., Schubnell, M., Sevilla-Noarbe, I., Smith, R. C., Soares-Santos, M., Sobreira, F., Suchyta, E., Swanson, M. E. C., Tarle, G., Thaler, J., Tucker, D., Walker, A. R., Wechsler, R. H., Zhang, Y., Dec. 2015. Discovery of two gravitationally lensed quasars in the Dark Energy Survey. *MNRAS* 454, 1260–1265.
- Akeret, J., Seehars, S., Amara, A., Refregier, A., Csillaghy, A., Mar. 2013. CosmoHammer: Cosmological parameter estimation with the MCMC Hammer. *Astrophysics Source Code Library*.
- Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., Greenfield, P., Droettboom, M., Bray, E., Aldcroft, T., Davis, M., Ginsburg, A., Price-Whelan, A. M., Kerzendorf, W. E., Conley, A., Crighton, N., Barbary, K., Muna, D., Ferguson, H., Grollier, F., Parikh, M. M., Nair, P. H., Unther, H. M., Deil, C., Woillez, J., Conseil, S., Kramer, R., Turner, J. E. H., Singer, L., Fox, R., Weaver, B. A., Zabalza, V., Edwards, Z. I., Azalee Bostroem, K., Burke, D. J., Casey, A. R., Crawford, S. M., Dencheva, N.,

²²<https://github.com/owertz/pySPT>

- Ely, J., Jenness, T., Labrie, K., Lim, P. L., Pierfederici, F., Pontzen, A., Ptak, A., Refsdal, B., Servillat, M., Streicher, O., Oct. 2013. Astropy: A community Python package for astronomy. *A&A*558, A33.
- Birrer, S., Amara, A., Refregier, A., Nov. 2015. Gravitational Lens Modeling with Basis Sets. *ApJ*813, 102.
- Birrer, S., Amara, A., Refregier, A., Aug. 2016. The mass-sheet degeneracy and time-delay cosmography: analysis of the strong lens RXJ1131-1231. *J. Cosmology Astropart. Phys.*8, 020.
- Birrer, S., Amara, A., Refregier, A., May 2017a. Lensing substructure quantification in RXJ1131-1231: a 2 keV lower bound on dark matter thermal relic mass. *J. Cosmology Astropart. Phys.*5, 037.
- Birrer, S., Refregier, A., Amara, A., Jan. 2018. Cosmic Shear with Einstein Rings. *ApJ*852, L14.
- Birrer, S., Welschen, C., Amara, A., Refregier, A., Apr. 2017b. Line-of-sight effects in strong lensing: putting theory into practice. *J. Cosmology Astropart. Phys.*4, 049.
- Bonvin, V., Courbin, F., Suyu, S. H., Marshall, P. J., Rusu, C. E., Sluse, D., Tewes, M., Wong, K. C., Collett, T., Fassnacht, C. D., Treu, T., Auger, M. W., Hilbert, S., Koopmans, L. V. E., Meylan, G., Rumbaugh, N., Sonnenfeld, A., Spiniello, C., Mar. 2017. H0LiCOW - V. New COSMOGRAIL time delays of HE 0435-1223: H_0 to 3.8 per cent precision from strong lensing in a flat Λ CDM model. *MNRAS*465, 4914–4930.
- Cappellari, M., Jun. 2002. Efficient multi-Gaussian expansion of galaxies. *MNRAS*333, 400–410.
- Collett, T. E., Auger, M. W., Sep. 2014. Cosmological constraints from the double source plane lens SDSSJ0946+1006. *MNRAS*443, 969–976.
- Dalal, N., Kochanek, C. S., Jun. 2002. Direct Detection of Cold Dark Matter Substructure. *ApJ*572, 25–33.
- Ding, X., Treu, T., Shajib, A. J., Xu, D., Chen, G. C.-F., More, A., Despali, G., Frigo, M., Fassnacht, C. D., Gilman, D., Hilbert, S., Marshall, P. J., Sluse, D., Vegetti, S., Jan. 2018. Time Delay Lens Modeling Challenge: I. Experimental Design. *ArXiv e-prints*.

- Eberhart, R., Kennedy, J., 01 1995. A new optimizer using particle swarm theory. ... of the sixth international symposium on
URL http://www.ppgia.pucpr.br/~alceu/mestrado/aula3/PSO_2.pdf
- Foreman-Mackey, D., Hogg, D. W., Lang, D., Goodman, J., Mar. 2013. emcee: The MCMC Hammer. PASP125, 306.
- Gavazzi, R., Treu, T., Koopmans, L. V. E., Bolton, A. S., Moustakas, L. A., Burles, S., Marshall, P. J., Apr. 2008. The Sloan Lens ACS Survey. VI. Discovery and Analysis of a Double Einstein Ring. ApJ677, 1046–1059.
- Gilman, D., Birrer, S., Treu, T., Keeton, C. R., Dec. 2017. Probing the nature of dark matter by forward modeling flux ratios in strong gravitational lenses. ArXiv e-prints.
- Grillo, C., Rosati, P., Suyu, S. H., Balestra, I., Caminha, G. B., Halkola, A., Kelly, P. L., Lombardi, M., Mercurio, A., Rodney, S. A., Treu, T., Feb. 2018. Measuring the value of the Hubble constant “‘a la Refsdal”. ArXiv e-prints.
- Hezaveh, Y., Dalal, N., Marrone, D., Mao, Y., Morningstar, W., Wen, D., Blandford, R., Carlstrom, J., Fassnacht, C., Holder, G., Kembell, A., Marshall, P., Murray, N., Perreault Levasseur, L., Vieira, J., Wechsler, R., 05 2016. Detection of lensing substructure using alma observations of the dusty galaxy sdg.81. The Astrophysical Journal 823, 37.
- Hezaveh, Y. D., Levasseur, L. P., Marshall, P. J., Aug. 2017. Fast automated analysis of strong gravitational lenses with convolutional neural networks. Nature548, 555–557.
- Hunter, J. D., 2007. Matplotlib: A 2d graphics environment. Computing In Science & Engineering 9 (3), 90–95.
- Jacobs, C., Glazebrook, K., Collett, T., More, A., McCarthy, C., Oct. 2017. Finding strong lenses in CFHTLS using convolutional neural networks. MNRAS471, 167–181.
- Keeton, C. R., Feb. 2011. GRAVLENS: Computational Methods for Gravitational Lensing. Astrophysics Source Code Library.

- Kelly, P. L., Diego, J. M., Rodney, S., Kaiser, N., Broadhurst, T., Zitrin, A., Treu, T., Perez-Gonzalez, P. G., Morishita, T., Jauzac, M., Selsing, J., Oguri, M., Pueyo, L., Ross, T. W., Filippenko, A. V., Smith, N., Hjorth, J., Cenko, S. B., Wang, X., Howell, D. A., Richard, J., Frye, B. L., Jha, S. W., Foley, R. J., Norman, C., Bradac, M., Zheng, W., Brammer, G., Molino Benito, A., Cava, A., Christensen, L., de Mink, S. E., Graur, O., Grillo, C., Kawamata, R., Kneib, J.-P., Matheson, T., McCully, C., Nonino, M., Perez-Fournon, I., Riess, A. G., Rosati, P., Borello Schmidt, K., Sharon, K., Weiner, B. J., Jun. 2017. An individual star at redshift 1.5 extremely magnified by a galaxy-cluster lens. ArXiv e-prints.
- Kneib, J.-P., Bonnet, H., Golse, G., Sand, D., Jullo, E., Marshall, P., Feb. 2011. LENSTOOL: A Gravitational Lensing Software for Modeling Mass Distribution of Galaxies and Clusters (strong and weak regime). Astrophysics Source Code Library.
- Lemon, C. A., Auger, M. W., McMahon, R. G., Ostrovski, F., Mar. 2018. Gravitationally Lensed Quasars in Gaia: II. Discovery of 24 Lensed Quasars. ArXiv e-prints.
- Lin, H., Buckley-Geer, E., Agnello, A., Ostrovski, F., McMahon, R. G., Nord, B., Kuropatkin, N., Tucker, D. L., Treu, T., Chan, J. H. H., Suyu, S. H., Diehl, H. T., Collett, T., Gill, M. S. S., More, A., Amara, A., Auger, M. W., Courbin, F., Fassnacht, C. D., Frieman, J., Marshall, P. J., Meylan, G., Rusu, C. E., Abbott, T. M. C., Abdalla, F. B., Allam, S., Banerji, M., Bechtol, K., Benoit-Lévy, A., Bertin, E., Brooks, D., Burke, D. L., Carnero Rosell, A., Carrasco Kind, M., Carretero, J., Castander, F. J., Crocce, M., D’Andrea, C. B., da Costa, L. N., Desai, S., Dietrich, J. P., Eifler, T. F., Finley, D. A., Flaugher, B., Fosalba, P., García-Bellido, J., Gaztanaga, E., Gerdes, D. W., Goldstein, D. A., Gruen, D., Gruendl, R. A., Gschwend, J., Gutierrez, G., Honscheid, K., James, D. J., Kuehn, K., Lahav, O., Li, T. S., Lima, M., Maia, M. A. G., March, M., Marshall, J. L., Martini, P., Melchior, P., Menanteau, F., Miquel, R., Ogando, R. L. C., Plazas, A. A., Romer, A. K., Sanchez, E., Schindler, R., Schubnell, M., Sevilla-Noarbe, I., Smith, M., Smith, R. C., Sobreira, F., Suchyta, E., Swanson, M. E. C., Tarle, G., Thomas, D., Walker, A. R., DES Collaboration, Apr. 2017. Discovery of the Lensed Quasar System DES J0408-5354. *ApJ* 838, L15.

- Mamon, G. A., Łokas, E. L., Nov. 2005. Dark matter in elliptical galaxies - II. Estimating the mass within the virial radius. *MNRAS*363, 705–722.
- Mao, S., Schneider, P., Apr. 1998. Evidence for substructure in lens galaxies? *MNRAS*295, 587.
- Metcalf, R. B., Madau, P., Dec. 2001. Compound Gravitational Lensing as a Probe of Dark Matter Substructure within Galaxy Halos. *ApJ*563, 9–20.
- Nierenberg, A. M., Treu, T., Brammer, G., Peter, A. H. G., Fassnacht, C. D., Keeton, C. R., Kochanek, C. S., Schmidt, K. B., Sluse, D., Wright, S. A., Oct. 2017. Probing dark matter substructure in the gravitational lens HE 0435-1223 with the WFC3 grism. *MNRAS*471, 2224–2236.
- Nierenberg, A. M., Treu, T., Wright, S. A., Fassnacht, C. D., Auger, M. W., Aug. 2014. Detection of substructure with adaptive optics integral field spectroscopy of the gravitational lens B1422+231. *MNRAS*442, 2434–2445.
- Nightingale, J., Dye, S., Massey, R., Aug. 2017. **AutoLens**: Automated Modeling of a Strong Lens’s Light, Mass and Source. ArXiv e-prints.
- Nightingale, J. W., Dye, S., Sep. 2015. Adaptive semi-linear inversion of strong gravitational lens imaging. *MNRAS*452, 2940–2959.
- Nord, B., Buckley-Geer, E., Lin, H., Diehl, H. T., Helsby, J., Kuropatkin, N., Amara, A., Collett, T., Allam, S., Caminha, G. B., De Bom, C., Desai, S., Dúmet-Montoya, H., Pereira, M. E. d. S., Finley, D. A., Flaugh, B., Furlanetto, C., Gaitsch, H., Gill, M., Merritt, K. W., More, A., Tucker, D., Saro, A., Rykoff, E. S., Rozo, E., Birrer, S., Abdalla, F. B., Agnello, A., Auger, M., Brunner, R. J., Carrasco Kind, M., Castander, F. J., Cunha, C. E., da Costa, L. N., Foley, R. J., Gerdes, D. W., Glazebrook, K., Gschwend, J., Hartley, W., Kessler, R., Lagattuta, D., Lewis, G., Maia, M. A. G., Makler, M., Menanteau, F., Niernberg, A., Scolnic, D., Vieira, J. D., Gramillano, R., Abbott, T. M. C., Banerji, M., Benoit-Lévy, A., Brooks, D., Burke, D. L., Capozzi, D., Carnero Rosell, A., Carretero, J., D’Andrea, C. B., Dietrich, J. P., Doel, P., Evrard, A. E., Frieman, J., Gaztanaga, E., Gruen, D., Honscheid, K., James, D. J., Kuehn, K., Li, T. S., Lima, M., Marshall, J. L., Martini, P., Melchior, P., Miquel, R., Neilsen, E., Nichol, R. C., Ogando, R., Plazas, A. A., Romer, A. K., Sako, M., Sanchez, E., Scarpine, V., Schubnell, M., Sevilla-Noarbe, I., Smith, R. C.,

- Soares-Santos, M., Sobreira, F., Suchyta, E., Swanson, M. E. C., Tarle, G., Thaler, J., Walker, A. R., Wester, W., Zhang, Y., DES Collaboration, Aug. 2016. Observation and Confirmation of Six Strong-lensing Systems in the Dark Energy Survey Science Verification Data. *ApJ*827, 51.
- Oguri, M., Oct. 2010. glafic: Software Package for Analyzing Gravitational Lensing. Astrophysics Source Code Library.
- Ostrovski, F., McMahon, R. G., Connolly, A. J., Lemon, C. A., Auger, M. W., Banerji, M., Hung, J. M., Koposov, S. E., Lidman, C. E., Reed, S. L., Allam, S., Benoit-Lévy, A., Bertin, E., Brooks, D., Buckley-Geer, E., Carnero Rosell, A., Carrasco Kind, M., Carretero, J., Cunha, C. E., da Costa, L. N., Desai, S., Diehl, H. T., Dietrich, J. P., Evrard, A. E., Finley, D. A., Flaugher, B., Fosalba, P., Frieman, J., Gerdes, D. W., Goldstein, D. A., Gruen, D., Gruendl, R. A., Gutierrez, G., Honscheid, K., James, D. J., Kuehn, K., Kuropatkin, N., Lima, M., Lin, H., Maia, M. A. G., Marshall, J. L., Martini, P., Melchior, P., Miquel, R., Ogando, R., Plazas Malagón, A., Reil, K., Romer, K., Sanchez, E., Santiago, B., Scarpine, V., Sevilla-Noarbe, I., Soares-Santos, M., Sobreira, F., Suchyta, E., Tarle, G., Thomas, D., Tucker, D. L., Walker, A. R., Mar. 2017. VDES J2325-5229 a $z = 2.7$ gravitationally lensed quasar discovered using morphology-independent supervised machine learning. *MNRAS*465, 4325–4334.
- Peng, C. Y., Ho, L. C., Impey, C. D., Rix, H.-W., Jul. 2002. Detailed Structural Decomposition of Galaxy Images. *AJ*124, 266–293.
- Peng, C. Y., Ho, L. C., Impey, C. D., Rix, H.-W., Jun. 2010. Detailed Decomposition of Galaxy Images. II. Beyond Axisymmetric Models. *AJ*139, 2097–2129.
- Refregier, A., 01 2003. Shapelets - i. a method for image analysis. *Monthly Notice of the Royal Astronomical Society* 338, 35.
- Refsdal, S., 01 1964. On the possibility of determining hubble’s parameter and the masses of galaxies from the gravitational lens effect. *Monthly Notices of the Royal Astronomical Society* 128, 307.
- Rossum, G., 1995. Python reference manual. Tech. rep., Amsterdam, The Netherlands, The Netherlands.

- Saha, P., Williams, L. L. R., Feb. 2011. PixeLens: A Portable Modeler of Lensed Quasars. Astrophysics Source Code Library.
- Schechter, P. L., Bailyn, C. D., Barr, R., Barvainis, R., Becker, C. M., Bernstein, G. M., Blakeslee, J. P., Bus, S. J., Dressler, A., Falco, E. E., Fesen, R. A., Fischer, P., Gebhardt, K., Harmer, D., Hewitt, J. N., Hjorth, J., Hurt, T., Jaunsen, A. O., Mateo, M., Mehlert, D., Richstone, D. O., Sparke, L. S., Thorstensen, J. R., Tonry, J. L., Wegner, G., Willmarth, D. W., Worthey, G., Feb. 1997. The Quadruple Gravitational Lens PG 1115+080: Time Delays and Models. *ApJ*475, L85–L88.
- Schechter, P. L., Morgan, N. D., Chehade, B., Metcalfe, N., Shanks, T., McDonald, M., May 2017. First Lensed Quasar Systems from the VST-ATLAS Survey: One Quad, Two Doubles, and Two Pairs of Lensless Twins. *AJ*153, 219.
- Shajib, A. J., Treu, T., Agnello, A., Jan. 2018. Improving time-delay cosmography with spatially resolved kinematics. *MNRAS*473, 210–226.
- Suyu, S., Marshall, P., Auger, M., Hilbert, S., Blandford, R., Koopmans, L., Fassnacht, C., Treu, T., 03 2010. Dissecting the gravitational lens b1608+656. ii. precision measurements of the hubble constant, spatial curvature, and the dark energy equation of state. *The Astrophysical Journal* 711, 201.
- Suyu, S., Treu, T., Hilbert, S., Sonnenfeld, A., Auger, M., Blandford, R., Collett, T., Courbin, F., Fassnacht, C., Koopmans, L., Marshall, P., Meylan, G., Spiniello, C., Tewes, M., 06 2014. Cosmology from gravitational lens time delays and planck data. *The Astrophysical Journal Letters* 788, L35.
- Suyu, S. H., Chang, T.-C., Courbin, F., Okumura, T., Jan. 2018. Cosmological distance indicators. *ArXiv e-prints*.
- Suyu, S. H., Marshall, P. J., Hobson, M. P., Blandford, R. D., Sep. 2006. A Bayesian analysis of regularized source inversions in gravitational lensing. *MNRAS*371, 983–998.
- Tagore, A. S., Keeton, C. R., Nov. 2014. Statistical and systematic uncertainties in pixel-based source reconstruction algorithms for gravitational lensing. *MNRAS*445, 694–710.

- Tessore, N., Bellagamba, F., Metcalf, R. B., Dec. 2016. LENSED: a code for the forward reconstruction of lenses and sources from strong lensing observations. *MNRAS*463, 3115–3128.
- Treu, T., Koopmans, L. V. E., Dec. 2002. The internal structure of the lens PG1115+080: breaking degeneracies in the value of the Hubble constant. *MNRAS*337, L6–L10.
- Treu, T., Marshall, P. J., Jul. 2016. Time delay cosmography. *A&ARv*24, 11.
- Vegetti, S., Despali, G., Lovell, M. R., Enzi, W., Jan. 2018. Constraining sterile neutrino cosmologies with strong gravitational lensing observations at redshift $z \sim 0.2$. ArXiv e-prints.
- Vegetti, S., Koopmans, L., Bolton, A., Treu, T., Gavazzi, R., 11 2010. Detection of a dark substructure through gravitational imaging. *Monthly Notices of the Royal Astronomical Society* 408, 1969.
- Vegetti, S., Koopmans, L. V. E., Jan. 2009. Bayesian strong gravitational-lens modelling on adaptive grids: objective detection of mass substructure in Galaxies. *MNRAS*392, 945–963.
- Vegetti, S., Lagattuta, D., McKean, J., Auger, M., Fassnacht, C., Koopmans, L., 01 2012. Gravitational detection of a low-mass dark satellite galaxy at cosmological distance. *Nature* 481, 341.
- Wertz, O., Orthen, B., Jan. 2018. pySPT: a package dedicated to the source position transformation. ArXiv e-prints.
- Williams, P. R., Agnello, A., Treu, T., Abramson, L. E., Anguita, T., Apostolovski, Y., Chen, G. C.-F., Fassnacht, C. D., Hsueh, J.-W., Motta, V., Oldham, L., Rojas, K., Rus, C. E., Shajib, A. J., Wang, X., Jun. 2017. Discovery of three strongly lensed quasars in the Sloan Digital Sky Survey. ArXiv e-prints.
- Xu, D., Sluse, D., Gao, L., Wang, J., Frenk, C., Mao, S., Schneider, P., Springel, V., Mar. 2015. How well can cold dark matter substructures account for the observed radio flux-ratio anomalies. *MNRAS*447, 3189–3206.