Galaxy shapes of Light (GaLight): a 2D modeling of galaxy images

Xuheng Ding\textsuperscript{a,}, Simon Birrer\textsuperscript{b,}, Tommaso Treu\textsuperscript{d}, John D. Silverman\textsuperscript{a,e}

\textsuperscript{a}Kavli Institute for the Physics and Mathematics of the Universe, The University of Tokyo, Kashiwa, Japan 277-8583 (Kavli IPMU, WPI)
\textsuperscript{b}Kavli Institute for Particle Astrophysics and Cosmology and Department of Physics, Stanford University, Stanford, CA 94305, USA
\textsuperscript{c}SLAC National Accelerator Laboratory, Menlo Park, CA, 94025
\textsuperscript{d}Department of Physics and Astronomy, University of California, Los Angeles, CA, 90095-1547, USA
\textsuperscript{e}Department of Astronomy, School of Science, The University of Tokyo, 7-3-1 Hongo, Bunkyo, Tokyo 113-0033, Japan

\textsuperscript{a}email: xuheng.ding@ipmu.edu

Abstract

GaLight is a Python-based open-source package that can be used to perform two-dimensional model fitting of optical and near-infrared images to characterize the light distribution of galaxies with components including a disk, bulge, bar, and quasar. The decomposition of stellar components has been demonstrated in published studies of inactive galaxies and quasar host galaxies observed by the Hubble Space Telescope and Subaru’s Hyper Suprime-Cam. GaLight utilizes the image modeling capabilities of lenstronomy while redesigning the user interface for the analysis of large samples of extragalactic sources. The package is user-friendly with some automatic features such as determining the cutout size of the modeling frame, searching for PSF-stars in field-of-view, estimating the noise map of the data, identifying all the objects to set the initial model and associated parameters to fit them simultaneously. These features minimize the manpower and allow the automatic fitting tasks. The software is distributed under the MIT license. The source code, installation guidelines, and example notebooks code can be found at \url{https://galight.readthedocs.io/en/latest/}.

Keywords: methods: data analysis; techniques: image processing; galaxies: photometry

1. Introduction

Accurate surface photometry and morphology of galaxies are key to understanding the evolution of galaxies. Over the past few decades, the light distribution of galaxies has been successfully studied using analytic functions to estimate their size, luminosity, concentration, orientation and symmetry. Of all tested analytic models, the Sérisc model function\cite{1} is the most successful and commonly used, which provides an accurate description of the surface brightness of galaxies, ranging from exponential disks (i.e., late-type galaxies) to de Vaucouleurs’s profiles\cite{2} for describing early-type galaxies. The success of these analytic functions further allow us to analyze the light of a unique type of object, i.e., active galactic nuclei (AGN), which has been recognized to play an important role in the evolution galaxies and the growth of its central supermassive black hole (SMBH). While a central AGN can be bright, even above that of its host galaxy, we can still infer the photometry and morphology of the underlying host galaxy based on the use of analytic models and deblending techniques.

With current and upcoming large imaging surveys, the sample of galaxies with quality imaging is growing rapidly and will continue to do so with future facilities. Thus, there is a great demand to process such large numbers of galaxy images in an automated manner with efficiency and accuracy.

Here, we present a first public release of GaLight, an open source PYTHON software package to model galaxy images including those hosting AGN. GaLight uses routines available through lenstronomy\cite{3,4} that are further described below. GaLight adds a number of features to process large numbers of images from survey data. This tool has been used to model the galaxy images from Hubble Space Telescope (HST)\cite{5,6} and Subaru Hyper Suprime-Cam (HSC)\cite{7,8}, including a sample of \~{}1.5 million galaxies and \~{}5 thousand quasars profiling from the automatic features.

While other packages, such as Galfit\cite{9} and GIMD2D\cite{10}, have provided the community much needed capabilities for analyzing galaxy images, there are benefits available with lenstronomy which we aim to fully exploit. For example, the parameter estimation is based on a forward modeling approach with semi-linear inversion of the amplitude components using machinery that includes a particle swarm optimizer (PSO)\cite{11,12} and Markov Chain Monte Carlo (MCMC) error analysis by emcee\cite{13}. The ability to account for uncertainties on the point spread function (PSF) is also an important feature. Equally important, lenstronomy – an astropy-affiliated PYTHON tool for gravitational lensing image simulations and analyses – is an open source software package that can be easily installed and adapted to the user’s modeling needs due to its flexibility.

While functions provided by lenstronomy are suitable to model the light distribution of a galaxy, some additional work is needed to define the models, set the initial parameters and priors, implement masks for nearby objects, and assess specific requirements for each image cutout. As described here, the soft-
ware package GALIGHT can set all the necessary parameters and perform automatic modeling with much needed efficiency, particularly for the analysis of large samples. Below, we introduce the basic functions of GALIGHT and demonstrate its capabilities.

The paper is organized as follows. In Section 2 and 3, we present a brief overview of the package which is structured in three major classes. In Section 4, we provide detailed information with some example code and figures. We highlight features in the Appendix and a summary is given in Section 5. We refer the reader to a number of studies that have made use of GALIGHT [5, 6, 7, 8].

2. Package overview

GALIGHT is open source python-based [14] software with distribution granted through the MIT open source software license. The software relies on the PYTHON standard library and the opensource libraries including NUMPY [15], SCIPY [16], ASTROPY [17] [18], MATPLOTLIB [19] and PHOTUTILS [20]. In particular, the package LENSTRONOMY [13] is the work horse that performs the fitting process. GALIGHT is compatible with both PYTHON 2 and PYTHON 3. However, the latter version is recommended.

The public release version of GALIGHT is on PyPi and can be installed using pip:

$ pip install galight --user

We also distribute GALIGHT on GitHUb, while more stable versions released through Pypi. Documentation is available through sphinx and released on ReadtheDocs. To fully demonstrate the usage of this package, we provide several Jupyter notebooks with extended features.

GALIGHT helps the user to execute model fitting with ease based on, but not limited to, automated features as listed as follows:

1. Determining a suitable cutout size for the image frame.
   The cutout should have a minimal size that balances the need to reduce computing time and capture the extent of low surface brightness emission. The cutout will also keep neighboring galaxies away from the edge of the frame. This enables these galaxies to be properly modeled which may have outer light profiles that overlap with the central target of interest.

2. The PSF model is key for image (de-)convolution and accounting for an unresolved point source component (i.e., AGN and small unresolved components). The common practice is to manually select a star in the field of view (FOV) for characterizing the PSF. GALIGHT will search for all the PSF-stars in the FOV, report their full width at half maximum (FWHM), and allow the user to choose those suitable for determining the empirical PSF model.

3. If needed, a noise map can be generated based on the exposure time and the background noise level measured from empty regions.

4. Neighboring sources can be simultaneously modeled or easily be masked out.

5. By default, all the objects in the modeling frame will be detected by the photutils.detect_sources. Based on the detection results, the parameter settings for all these sources will be assigned automatically.

6. Output data products are generated for full assessment of the goodness-of-fit with the ability to share across different computing platforms.

In practice, GALIGHT is able to perform the fitting tasking without manual supervision after the target position and the modeling materials are provided.

3. Model-fitting procedure

GALIGHT uses three steps to achieve a successful modeling of image data. Here, we provide a brief overview of each step as defined by one PYTHON class and provide further details in Section 3.

1. DataProcess: The data products are prepared including the cutout of the science image, noise map, PSF model, and nearby object masks (optional). The image pixel scale and magnitude zero point should also be provided to GALIGHT to calculate physical quantities (i.e., target magnitudes and galaxy size in arcsec).

2. FittingSpecify: The model components (e.g., single Sérsic profile, disk-bulge, AGN, etc.) and associated parameters should be defined by this class. For example, the initial values and priors of the parameters are assigned. Priors can be given as either a range for each parameter with uniform or weighted sampling, or a single fixed value.

3. FittingProcess: The model fitting is performed, and the results are presented in this step. The user can determine the method of parameter inference (e.g., minimization and MCMC). Once the fitting has completed, this class provides the modeling output for diagnostic and demonstration purposes.

3.1. Analytical model and fitting strategies

Through LENSTRONOMY, the model components are defined and the fitting is performed. The imported features include the definition of the light profile and the approach to sampling the parameter space.

Light profiles: All extended light profiles defined by LENSTRONOMY are imported and available. These include, but not limited to, Sérsic [1], Chameleon [21], ellipsoid, 2D Gaussian, pseudo
Jaffe \cite{22}, Moffat \cite{23}, and power-law functions. We refer the reader to the LENSTRONY documentation for further details.

By default, the Sersic profile \cite{lenstronomy} is adopted to describe the light of extended object with parameterization as follows:

\[ I(R) = A \exp \left[ -k \left( \left( \frac{R}{R_{\text{eff}}} \right)^{1/n} - 1 \right) \right], \quad (1) \]

\[ R(x, y, q) = \sqrt{x^2 + y^2 / q^2}, \quad (2) \]

where \( A \) is the amplitude and \( q \) denotes the axis ratio. The Sersic index \( n \) controls the shape of the radial surface brightness profile, a larger \( n \) corresponds to a steeper inner profile and a highly extended outer wing. \( k \) is a constant which solely depends on \( n \) to ensure that the isophote at \( R = R_{\text{eff}} \) encloses half of the total light \cite{lenstronomy}. Note that this definition of Sersic profile is the same as the one adopted by GALFIT in which the \( R_{\text{eff}} \) is the semi-major-axis half-light radius. The other definition of elliptical light profile with a format as \( R(x, y, q) = \sqrt{q x^2 + y^2 / q^2} \) is also available in LENSTRONY after version 1.9.0; hence GALFIT. Both formats will be available to the user\footnote{Note that \( A \) is also the surface brightness at \( R_{\text{eff}} \).} however, note that the \( R_{\text{eff}} \) will be different by a factor of \( \sqrt{q} \) between these two definitions.

We note that the light profile for spheroidal galaxies can usually be fitted with a single Sersic profile and a free index \( n \). In cases where more components are required, a composite light model can be implemented, such as a spheroid plus disk component (Sersic \( n = 1 \)) and even a bar (Sersic \( n = 0.5 \)).

When modeling the point-source flux component, such as an AGN component or star, a scaled PSF model will be adopted to characterize the unresolved emission. The amplitude and the position of the point source will be considered as free parameters. As demonstrated in next section, a PSF model can be directly input to GALFIT. Alternatively, GALFIT can search for suitable stars over the full image to generate an empirical PSF model. To account for an imperfect model PSF, one can also provide an uncertainty on the PSF which will be considered when calculating the likelihood and best-fit model parameters. Taking the PSF model uncertainty into account can be important, especially when estimating the uncertainty of the fitting parameters. This is likely an important feature of GALFIT which is not a feature directly implemented in the currently most commonly used modeling codes. To note, the PSF model is also used for the convolution of the total emission in the forward-modeling.

**Fitting algorithm:** To obtain the inference of the best-fit parameters, a likelihood is calculated by LENSTRONY using the lenstronomy.ImSim module while sampling of the parameter space is performed by the lenstronomy.Sampling module. Briefly, a likelihood estimator is implemented to determine the values of the posterior distribution based on the data and model \( p(d_{\text{data}}|d_{\text{model}}) \) (i.e., the probability of the data given a model) using all the counted pixels in the full cutout which can be presented as follows:

\[ \log p(d_{\text{data}}|d_{\text{model}}) = \sum_i \frac{(d_{\text{data},i} - d_{\text{model},i})^2}{2\sigma_i^2} + C. \quad (3) \]

The errors (i.e., \( \sigma_i \)) are assigned based on the noise map. For fitting, the noise map can be directly input to GALFIT. If not provided, the tool can calculate the total noise map according to the following two terms:

\[ \sigma_i^2 = \sigma_{\text{bkgd}}^2 + |d_{\text{data},i}|/(t_{\text{exp}} \times G_{\text{eff}}), \quad (4) \]

i.e., a Gaussian background term (i.e., \( \sigma_{\text{bkgd}} \)), and a Poisson term, i.e., \( \sqrt{|d_{\text{data},i}|/(t_{\text{exp}} \times G_{\text{eff}})} \), where \( t_{\text{exp}} \) is the exposure time (in seconds) of the observational data (given in counts per second) and \( G_{\text{eff}} \) is the effective gain values. By default, the Sersic surface brightness amplitude (i.e., the parameter \( A \) in Equation \cite{lenstronomy}) is linear parameters that is solved directly during the likelihood estimation. This modeling strategy saves a type of non-linear parameter and would lead to better and faster convergence; even more, it can be generalized to more components and basis sets.

4. Learning to use GALFIT through examples

We provide detailed descriptions of the routines and three core classes. In this paper, we give brief show-case examples and further demonstrations can be found online through Jupyter notebook\footnote{https://github.com/dartoog/galight_notebooks}.

4.1. DataProcess

The following example in next page is a demonstration of the steps needed to prepare the class DataProcess. We provide example code (see next page) and highlight the relevant lines being discussed.

GALFIT retrieves the pixel scale from the image header (line 16), or this is set by the user manually data_process.deltaPix. The location of the target can be given as either a pixel position or with ‘WCS’ information as RA and DEC (line 11). The function generate_target_materials (line 19) extracts a square region centered on the target and creates an image stamp having a frame size of (2\text{radius+1}). If the radius value is assigned as ‘None’, GALFIT will determine a minimum value, ranging from 30 to 70 pixels, that avoids having nearby objects not fully covered by the cutout image. Of course, for nearby galaxies, the upper boundary might not be sufficient and the user should select their own value.

Objects in the frame are detected as part of the generate_target_materials function, and characterized by a set of elliptical apertures as shown in Figure\cite{lenstronomy}(left and middle
import astropy.io.fits as pyfits

dataFile = pyfits.open('../example_data.fits')
fov_image = dataFile[1].data
header = dataFile[1].header
noiseFile = pyfits.open('../example_noise.fits')
noise_map = noiseFile[1].data

#Prepare for the fitting#
from galight.data_process import DataProcess
pos_RA, pos_Dec = 500, 500 #Assuming the target is at pixel position 500, 500#
zp = 27.0 #Assuming the zeropoint of the detection is 27.0#
rm_bkglight = True #The background light will be modeled and removed as a 2D profile by

#Input the information to DataProcess.#
data_process = DataProcess(fov_image = fov_image, target_pos = [pos_RA, pos_Dec],
                          fov_noise_map = noise_map, header = header, rm_bkglight = rm_bkglight, zp = zp)

#Preparing fitting ingredients, including cutout image, noise map and masks (if needed).#
data_process.generate_target_materials(radius=65, create_mask = True, nsigma=2.8, exp_sz= 1.2,
                                         npixels = 15)

#Automatically find all the PSF-stars and select the best one to use for fitting.#
data_process.find_PSF(radius = 30, user_option = True)

#Check out if all the fitting ingredients are ready. If not, an error message will appear.#
data_process.checkout()
The apertures determined by DataProcess() in previous step are applied and the initial Sérsic parameters are set as starting points for the fitting. The main settings of the fit routine are controlled by prepare_fitting_seq() which take arguments that can be set as follows:

1. point_source_num sets the number of point sources needed for the fit (default = 0). This should be set to one for a quasar or a star. By default, the software will automatically find the source with brightest peak in the cutout frame and use the peak position as the initial position to perform the fit.
2. fix_center_list sets the priors on the position so that an AGN and its host galaxy are fixed to the same central position.
3. fix_n_list can be assigned to fix the Sérsic index value and fix_Re_list can be assigned to fix the Sérsic $R_{\text{eff}}$ value.
4. psf_error_map defines the PSF error map with the same dimensions as the input PSF. The values of the error map are added at the fitted point source position to the final noise map as:

$$\sigma_i^2 = \sigma_i^2 + \text{psf\_error\_map}\ast(\text{point\_source\_flux})^2.$$ 

Examples of these setting can be found in the online notebooks. Detailed information is given in the online documentation through ReadtheDocs.

The prepare_fitting_seq() is designed to help the user quickly build up the requisites for the fitting using the templates embedded in the function. After this step, the user can make modifications to these settings to add complexity and make the fitting more sophisticated. For example, the FittingSpecify.kwargs_params is a dictionary that saves all the parameters settings for the Sérsic model and the point sources including the initials, sampling steps, fixed value, lower and upper limit. The user can change the corresponding values therein to modify the lower and upper limits for any parameters so that these parameters will be limited in the desired range and sampled with a flat prior. Additional priors can also be added to the likelihood. For example, defining the disk size to be larger than the bulge size or any arbitrary prior on the parameters (e.g., a Gaussian prior), can be assigned through the FittingSpecify.kwargs_likelihood. These are demonstrated in "galight_HSC.galaxy.ipynb". Note that all this settings are in the format that defined by lenstronomy; the user should follow its guidance to achieve more sophisticated fittings.

Finally, the user can use plot_fitting_sets() to display how the target and nearby sources will be modeled as shown here in the right panel of Figure 1.

4.3. FittingProcess

The fitting routine is executed after FittingSpecify has been passed to FittingProcess:

```python
from galight.fitting_process import FittingProcess
fit_sepc = FittingProcess(data_process)
fit_sepc.plot_fitting_sets()
fit_sepc.build_fitting_seq()
```

Data
---

Segmentation Image
---

Data and components used to fit
fit_run = FittingProcess(fit_sepc, savename
return 'savename')
# Setting the fitting approach and run. The
fitting process will be save as
savepath = 'savename.pkl'
fit_run.run(algorithm_list=['PSO', 'MCMC',
setting_list=[None, None])

By default, the procedure first performs the minimization of the parameters using the particle swarm optimizer ('PSO, [11, 12]) algorithm and then passes the inferred minimized parameters to the MCMC routine (e.g., emcee [13]) to estimate their best-fit values and uncertainties. In the FittingProcess.run(), i.e., line 6 in the example, the user can re-define the algorithm_list to increase the times of minimization or skip the MCMC through the arguments. Moreover, the sampling parameters such as the fitting particles and the number of iterations can be defined through setting_list.

Once parameter estimation has finished, the trace of the PSO and MCMC fitting chains will be plotted to help the user diagnose the convergence of the fitting routine.

The MCMC parameter inferences are translated to fluxes values and displayed in the bottom panel of Figure 2 with the posterior distributions on the flux of the individual components.

5. Conclusion

We present an open-source python-based tool – galight – to decompose galaxies into their various components. The software is distributed under the MIT license. Taking advantage of the functionalities of lenstronomy, the modeling is composed of three steps, separated into individual classes. We briefly introduce the functions of each class and present the example results in Section 4. galight provides additional functions to setup the fitting routine and complete the modeling in an automated manner. Our aim is to minimize intervention by the user so that large data sets can be processed seamlessly.

The demonstrations and other features mentioned in this paper are included in example notebooks in https://github.com/dartoon/galight_notebooks as listed here.

1. 'galight_HST_QSO.ipynb': Modeling example of an HST observed quasar image. In this example, the PSF is reconstructed by a PSF-star in the FOV and the noise map is calculated using galight based on the approach introduced in Appendix A.

2. 'galight_HSC_QSO.ipynb': Modeling example of an HSC imaged quasar, in which the PSF and noise map are supplied to galight.

3. 'galight_HSC_dualAGN.ipynb': Modeling example of an HSC imaged double AGN candidate.

4. 'galight_HSC_galaxy.ipynb': Modeling example of an HSC imaged galaxy. As demonstration, the galaxy is modeled as composite model with disk and bulge components.

These online resources will be updated periodically.

Citing the code: This manuscript is posted on arXiv as a user manual, which will be constantly updated in the future with more upcoming features. We ask the users of galight please make reference to Ding et al., ApJ, 2020, 888, 37 and LENSTRONOMY [4] when use it.

6. Acknowledgements

This software is supported by World Premier International Research Center Initiative (WPI), MEXT, Japan.

Appendix A. Noise map estimation

If user does not provide an error map, galight will calculate the noise level for each pixel which is composed of a Gaussian background noise and Poisson noise components, (see Equation 4). First, galight detects all objects and then masks them to identify the empty (sky) regions to measure the average signal. We demonstrate an example of this measurement in Figure A.3. A Gaussian formalization is used to measure the standard derivation. Since the background light has been subtracted by setting rm_bkglight = True in DataProcess(), the median of the distribution should close to zero, which can be easily checked (Figure A.3 right panel). The Poisson noise is estimated based on the pixel number counts and the exposure time, where exposure time can be input as either a single value or a array map. Alternatively, the user can measure the Gaussian background themselves and directly input the error map to galight.

Appendix B. Search for PSF-stars in the image FOV

galight can search for available PSF-stars in the FOV of the image using function DataProcess.findPSF. The algorithm is designed so that it looks for the local maximum pixel with prominent features (like PSF), achieved by the function scipy.ndimage.filters. During this search process, the code measures the FWHM values of each PSF candidates and discards those with extended profiles. Moreover, PSF-stars that are too faint or bright, compared with the target will be removed by default.
Figure 2: **top panels**: Example modeling results of the image data shown in Figure 1. The panels from the left to right are: (1) observed science data, (2) best-fit model. The clear box around the galaxy reveals the region of the PSF size that used to model the AGN. (3) data - point-source model (i.e., host galaxy only) (4) residuals divided by variance (i.e., reduced $\chi^2$ map), (5) surface brightness profiles in 1D (top) and the residual (bottom). **Bottom panels**: the MCMC inference of the likelihood distribution of the fitted parameters are translated to flux values (i.e., the total pixel counts) for each fitted component. The labels for each component correspond to those in the right panel of Figure 1. Note that the flux values are not the parameters that fitted directly.
We demonstrate the search of PSF stars in the FOV. The FWHM and 1D profile is also presented to the user to aid in the selection of the best PSF to use for the fitting (see Figures B.4, B.5, and B.6). By default, the PSF with the sharpest profile (i.e., the one with smallest FWHM value) will be selected.

References

[1] J. L. Sersic, Atlas de Galaxias Australes, 1968.
[2] G. de Vaucouleurs, Recherches sur les Nebuleuses Extragalactiques, Annales d’Astrophysique 11 (1948) 247.
[3] S. Birrer, A. Amara, lenstronomy: Multi-purpose gravitational lens modelling software package, Physics of the Dark Universe 22 (2019) 189–201.
[4] S. Birrer, A. Shajib, D. Gilman, A. Galan, J. Aalbers, M. Millon, R. Morris, G. Pagoni, J. Park, L. Tedori, N. Tessore, M. Ueland, L. Van de Vyvere, S. Wagner-Carena, E. Wenpe, L. Yang, X. Ding, T. Schmidt, D. Sluse, M. Zhang, A. Amara, lenstronomy II: A gravitational lensing software ecosystem, The Journal of Open Source Software 6 (2021) 002.
[5] S. Birrer, A. Shajib, D. Gilman, A. Galan, J. Aalbers, M. Millon, R. Morris, G. Pagoni, J. Park, L. Tedori, N. Tessore, M. Ueland, L. Van de Vyvere, S. Wagner-Carena, E. Wenpe, L. Yang, X. Ding, T. Schmidt, D. Sluse, M. Zhang, A. Amara, lenstronomy II: A gravitational lensing software ecosystem, The Journal of Open Source Software 6 (2021) 002.
[6] V. N. Bennett, T. Treu, X. Ding, J. Stomberg, S. Birrer, T. Snyder, M. A. Malkan, A. W. Stephens, M. W. Auger, A local baseline of the black hole mass scaling relations for active galaxies. IV. Correlations between $M_{\text{BH}}$ and host galaxy $\sigma$, stellar mass, and luminosity, arXiv e-prints (2021) arXiv:2101.03555.
[7] S. Tang, J. D. Silverman, X. Ding, J. Li, K.-G. Lee, M. A. Strauss, A. Goulding, M. Schrann, L. Kawaiwanchikajj, J. X. Prochaska, J. F. Hennawi, M. Imarishi, K. Iwasawa, Y. Toba, I. Kayo, M. Oguri, Y. Matsuoka, K. Ichikawa, T. Hartwig, N. Kusakawa, T. Kawaguchi, K. Kohno, Y. Matsuda, T. Nagao, Y. Omo, M. Onoue, M. Ouchi, K. Shimasaku, H. Suh, N. Suzuki, Y. Taniguchi, Y. Ueda, N. Yasuda, Optical Spectroscopy of Dual Quasar Candidates from the Subaru HSC-SSP program, arXiv e-prints (2021) arXiv:2105.10183.
[8] J. Li, J. D. Silverman, X. Ding, M. A. Strauss, A. Goulding, S. Birrer, H. M. Yusuf, Y. Xue, L. Kawaiwanchikajj, Y. Matsuoka, Y. Toba, T. Nagao, M. Schrann, K. Inayoshi, The Sizes of Quasar Host Galaxies with the Hyper Suprime-Cam Subaru Strategic Program, arXiv e-prints (2021) arXiv:2105.06566.
[9] C. Y. Peng, L. C. Ho, C. D. Impey, H.-W. Rix, Detailed Structural Decomposition of Galaxy Images, AJ 124 (1) (2002) 266–293.
[10] L. Simon, GIM2D: an IRAF package for the Quantitative Morphology Analysis of Distant Galaxies, in: R. Albrecht, R. N. Hook, A. A. Bushouse (Eds.), Astronomical Data Analysis Software and Systems VII, Vol. 145 of Astronomical Society of the Pacific Conference Series, 1998, p. 108.
Figure B.4: Identifying suitable PSF stars in a HST/WFC3 field. Candidates found by \texttt{galight} are marked by small blue boxes and checked as to whether they are not too bright or too faint.

Figure B.5: Zoom-in images of the PSF model candidates. During this process, the FWHM (in units of pixel scales) of each PSF is measured and reported to help the user select the best PSFs for their application.
Figure B.6: The 1D surface brightness profile of each PSF model candidates compared to the targets.