Preprint typeset using LaTeX style emulateapj v. 08/13/06

PYGFIT: A TOOL FOR EXTRACTING PSF MATCHED PHOTOMETRY

Conor L. Mancone[1], Anthony H. Gonzalez[1], Leonidas A. Moustakas[2], Andrew Price[2]

ABSTRACT

We present PyGFit, a program designed to measure PSF-matched photometry from images with disparate pixel scales and PSF sizes. While PyGFit has a number of uses, its primary purpose is to extract robust spectral energy distributions (SEDs) from crowded images. It does this by fitting blended sources in crowded, low resolution images with models generated from a higher resolution image. This approach minimizes the impact of crowding and also yields consistently measured fluxes in different filters, minimizing systematic uncertainty in the final SEDs. PyGFit currently supports point source and Sérsic models. We present an example of applying PyGFit to real data and perform simulations to test its fidelity. Care must be taken when constructing the high resolution catalog, as any missing information introduces an important source of error into PyGFit. The uncertainty in the fitted flux rises sharply as a function of nearest-neighbor distance for objects with a neighbor within 60% of the PSF size. Similarly, the uncertainty increases quickly for objects blended with a neighbor $\gtrsim 4$ times brighter. For all other objects the fidelity of PyGFit's results depends only on flux, and the uncertainty is primarily limited by sky noise.

Subject headings:

1. INTRODUCTION

Astronomy has increasingly benefited from high-resolution imaging, exemplified by the Hubble Space Telescope, with a point-spread function (PSF) size of $< 0.1''$ (Dressel 2011). Obtaining ancillary multi wavelength data at comparable resolution is often impractical, and it is commonly necessary to work with mixed resolution data sets. For instance, lower resolution ground-based data are often used in conjunction with high resolution space-based imaging, and at near-infrared wavelengths higher-resolution imaging is often not available or feasible. In such cases, effective crowding can vary substantially as a function of wavelength, and the quality of the final data set is limited by the reliability of fluxes extracted from the most crowded images. So long as sources remain unresolved, PSF fitting provides a viable method for extracting magnitudes in crowded fields. However, a different procedure is needed to measure magnitudes of resolved or marginally-resolved sources in crowded fields. With mixed-resolution datasets, such a procedure must measure magnitudes in a consistent way despite differences in PSF, resolution, and crowding.

We present a new program, Python Galaxy Fitter (PyGFit) aimed at solving these problems. PyGFit is not the first program to address these issues (see for example Fernández-Soto et al. 1999, Labbé et al. 2005, Laidler et al. 2007, and de Santis et al. 2007). Indeed, PyGFit and the well-known codes TFIT (Laidler et al. 2007) and ConvPhot (de Santis et al. 2007) are conceptually similar. TFIT and ConvPhot work by taking cutouts from a high-resolution image (HRI), convolving them with the low-resolution PSF, and fitting these models directly to the low-resolution image (LRI). As a result, the HRI and LRI must be astrometrically aligned, and their pixels have to be properly matched. In the case of ConvPhot the pixel scale must be the same in the low- and high-resolution images, and any offsets between the images must be integer pixel offsets and have to be fed into the program. In the case of TFIT, the pixel scale of the LRI must be an integer factor of the pixel scale of the HRI, and the images must cover exactly the same area on the sky. In both cases any sub-pixel offsets between the two images can introduce errors into the final results (Laidler et al. 2007).

PyGFit, however, uses analytic source models. It works on the basis of a high-resolution catalog (HRC) which gives the parameters of a model fit (i.e. a Sérsic profile) for every object in the HRI. PyGFit fits those models to the LRI, simultaneously fitting blended sources. The use of models minimizes the impact of shot-noise in the HRI, especially for objects with low S/N ratio. Also, this decouples the HRI and LRI, such that the LRI can have an arbitrary pixel scale and can cover larger or smaller areas on the sky. Surveys with high resolution imaging routinely fit model profiles to all visible sources, which means that PyGFit can often build off of already existing catalogs. Moreover, PyGFit is relatively fast, in many cases taking just a few minutes to fit the LRI for an area of the sky corresponding to a single *HST* pointing. PyGFit performs an alignment step to account for any zeroth order offsets between the WCS of the HRI and LRI, and can also account for small sub-pixel shifts between the two images. PyGFit currently supports two models, a point source and Sérsic model, and is extensible to include any analytic profile. It also has a built-in capability to quantify uncertainties via simulations of artificial galaxies.

The intended purpose of PyGFit is to measure PSF-matched photometry from mixed-resolution datasets, especially for marginally-resolved sources in crowded fields. This enables reliable measurements of galaxy SEDs and consequently, stellar mass fits. However, PyGFit is not limited to this single application. As a profile fitting routine, it has a number of other potential uses. For instance, PyGFit can be used to subtract foreground

Electronic address: cmancone@astro.ufl.edu
[1] Department of Astronomy, University of Florida, Gainesville, FL 32611
[2] Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109

sources from an image to search for faint background sources (such as gravitational arcs). It can also subtract objects identified in one image from another image (presumably taken at a different wavelength), a feature that can be used, for instance, to identify high-z dropout candidates.

This paper is structured as follows. Section 2 describes PyGFit's fitting procedure. Section 3 demonstrates PyGFit's usage on real data and discusses some relevant limitations. Section 4 describes the simulations built into PyGFit and uses them to measure the fidelity of PyGFit. Finally, Section 5 gives our conclusions. All magnitudes are on the Vega system, and we assume a WMAP 7 cosmology (Komatsu et al. 2011; $\Omega_m = 0.272, \Omega_\Lambda = 0.728, h = 0.704$) throughout.

2. PROCEDURE

2.1. *Overview*

The fundamental goal of PyGFit is to enable matched photometry in mixed-resolution data sets that is robust to the effects of crowding in the lower resolution images. In the limiting case where a source is effectively a point source in the lower resolution data set, this problem has long been solved as it is effectively a matrix inversion process (see e.g. the MOPEX software for MIPS photometry; Makovoz & Marleau 2005, or DAOPHOT; **?**). Optimal deblending however becomes more challenging and computationally intensive when sources are even marginally-resolved, and the convolution of the PSF and underlying galaxy profile must be considered.

With PyGFit we present an approach that is designed to be fast, flexible, and reliable. This code was originally designed to enable such robust photometry in the crowded cores of high-redshift cluster galaxies using the combination of HST and Spitzer data, but is generally applicable to any situation in which one desires profile-matched photometry between mixed resolution data sets. As described below, PyGFit can successfully deblend the photometry of two sources as long as their intrinsic separation is more than approximately 60% of the FWHM of the PSF in the low resolution data. It is also important to note that PyGFit makes the implicit assumption that the shape of the underlying profile is the same at all wavelengths − effectively an assumption that morphological $k-$corrections are small. In cases where the morphology changes strongly with wavelength, such as a starburst galaxy with an underlying old stellar population, the results from PyGFit should be treated with care. Such cases may be flagged through the galaxies' colors and SEDs.

At its core, PyGFit uses position and shape information of objects in a high-resolution image (HRI) to determine how to divide the luminosity of overlapping objects in a low-resolution image (LRI) among the constituent components. As such, the primary input into PyGFit is a high-resolution catalog (HRC) that gives positions and shapes of all objects in the HRI. PyGFit's procedure can be broadly separated into four steps: object detection and segmentation of the LRI, alignment of the HRC with the LRI, object fitting, and final catalog generation. There are five primary inputs into PyGFit which must be provided: the HRC, the LRI, an RMS map for the LRI, the PSF image of the LRI, and a Source Extractor configuration file for the LRI.

The HRC should give Sérsic model parameters for all objects in the HRI which are resolved in the LRI. This requires fitting a Sérsic profile to every object in the HRI, a task which is becoming common for surveys with *HST* imaging. While any program can be used to fit models to the HRI, the modeling routines built into PyGFit use precisely the same equations as GALFIT (Peng et al. 2002, 2010), enabling the output from GALFIT to be fed directly into PyGFit. Therefore, the simplest way to build the HRC is by using a program that can run GALFIT and fit a Sérsic profile to every object in the image (for example GALAPAGOS, Häußler et al. 2011).

The first step PyGFit executes, object detection and segmentation, is performed by running Source Extractor (Bertin & Arnouts 1996) on the LRI. The primary goal of this step is to generate a segmentation map of the LRI. This provides a convenient method for determining which objects in the HRC are blended together and hence must be modeled together, and it also divides the process into manageable chunks. Source Extractor also creates a low-resolution catalog (LRC) and a background map. PyGFit stores the LRC and includes any desired information from it in the final output catalog. The background map is used to estimate the sky for all objects, and is subtracted from the LRI before fitting.

This is followed by an alignment step between the HRC and the LRI which serves two purposes. First, it accounts for any zeroth order offsets between the WCS of the HRC and the LRI. Next, it accounts for any miscentering of the low-resolution PSF image. PyGFit performs this global alignment by finding isolated objects and calculating the offset via least squares minimization. PyGFit takes the median fitted position offset and then adjusts the positions of objects in the HRC accordingly.

PyGFit then moves on to fitting all the objects. It iterates through the segmentation regions of the LRI (i.e. the low-resolution sources) and finds all overlapping objects from the HRC. PyGFit then generates and stores a model for all matching objects from the HRC, convolves each with the low-resolution PSF as needed, and performs a χ^2 minimization using a Levenberg−Marquardt algorithm to fit the models to the low-resolution source. During the χ^2 minimization only the positions and fluxes of the objects are left as free parameters. All other Sérsic parameters (radius, Sérsic index, aspect ratio, and position angle) are held fixed. The positions are restricted to small shifts (typically less than a pixel) and the fluxes are constrained to be positive.

Finally, the output catalog is generated. This consists of the final fluxes measured by PyGFit for the objects in the HRC, any information requested from the LRC, and various diagnostics of each object. At this stage PyGFit also generates a residual image for quick quality control and assessment. Figure 1 gives a high level overview of PyGFit's procedure, showing the primary inputs required by PyGFit on the top, the main steps it executes, and how the various inputs feed into each step.

2.2. *Object Detection and Segmentation*

The first thing PyGFit does is to run Source Extractor on the low resolution. It feeds the RMS map into Source Extractor and detection limits are set in a Source Extractor configuration file. PyGFit uses three data products

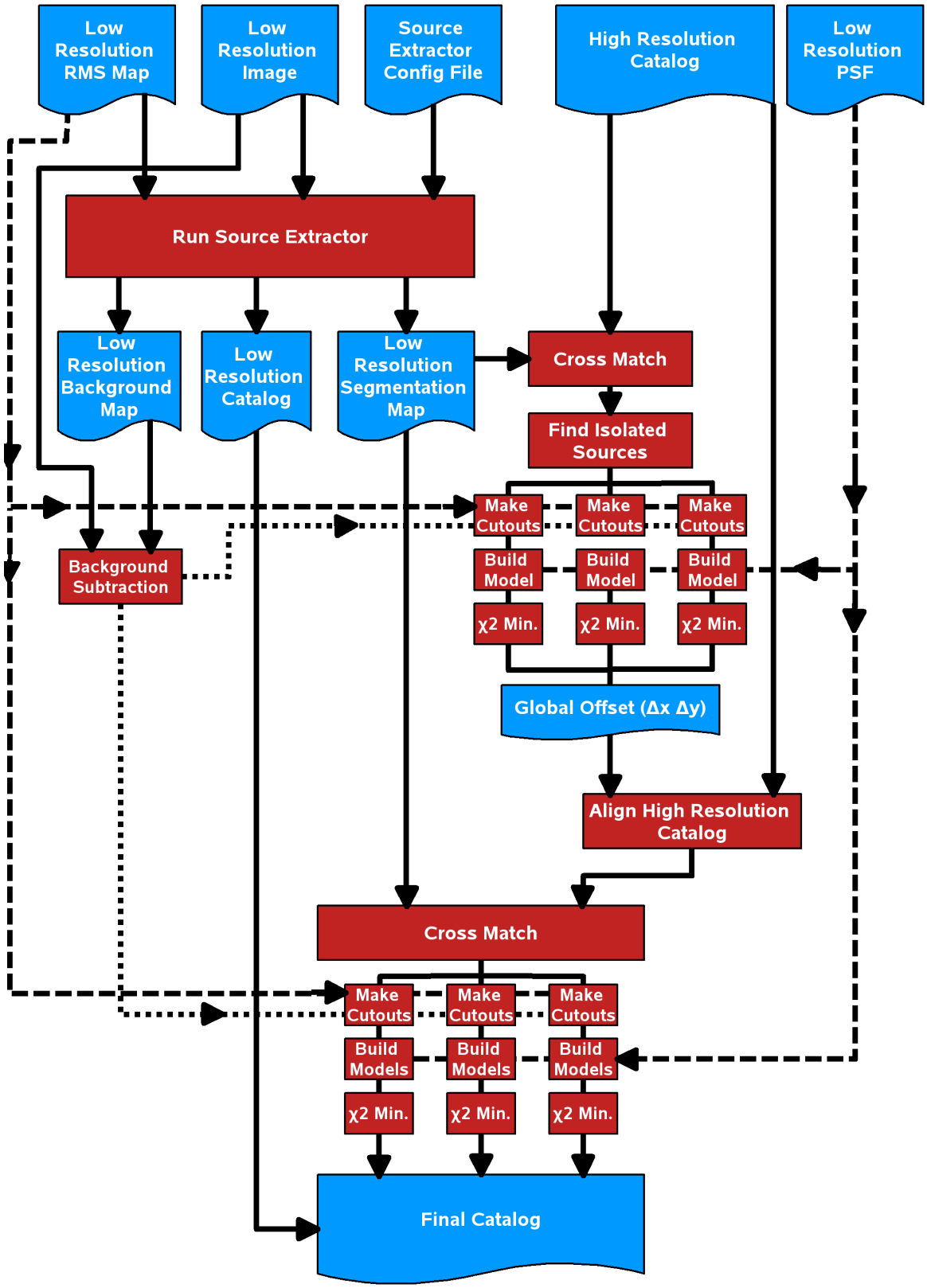

FIG. 1.— A flow chart of PyGFit's procedure. Rectangles denote computational processes executed by PyGFit while the page symbols denote data products created by or used by PyGFit. The five data products along the top are inputs which must be provided to PyGFit.

from this Source Extractor run: the segmentation map, the LRC, and the background map. The most important output from Source Extractor is the segmentation map, which PyGFit uses to separate the process into distinct chunks. We refer to each region of the segmentation map as a low resolution source. A single low resolution source can have any number of objects from the HRC associated with it. In practice, PyGFit ignores all low resolution sources which do not have any overlapping objects from the HRC.

One advantage of Source Extractor is that it has an easily configurable level of deblending. It is obviously preferable to minimize the amount of deblending done by Source Extractor since deblending is the very purpose behind PyGFit. However, a complete lack of deblending on Source Extractor's part can result in large segments of the LRI being assigned to one low resolution source. This in turn can cause unreasonably large execution times as PyGFit attempts to perform χ^2 minimization for a problem with hundreds of free parameters. In such cases allowing Source Extractor to perform a small amount of deblending can dramatically decrease execution with little to no loss of fidelity for the final results.

Source Extractor also generates a LRC which PyGFit stores. PyGFit does not use any of the information in the LRC but simply passes it along to the final catalog. By default, PyGFit extracts positions and auto-magnitudes from the LRC and copies them into the final catalog. However, it can also pass along additional parameters from Source Extractor if desired.

Finally, PyGFit subtracts the background map from the LRI to remove its contribution from the low-resolution sources.

2.3. *Catalog Alignment*

Next PyGFit runs an alignment step. The alignment step accounts for any zeroth-order misalignment of the HRC and LRI, as well as any miscentering of the low resolution PSF image. PyGFit begins the alignment step by identifying isolated sources, i.e. low resolution sources which only have one associated object from the HRC. PyGFit has configurable parameters to determine how many isolated sources should be used for the alignment step, as well as to limit them to a particular magnitude range.

After identifying isolated sources, PyGFit fits them using its normal routine (Section 2.4) but with a larger allowed position shift than during normal fitting. The precise size of the allowed position offset is configurable by the user, and should be large enough to account for any potential offset between the HRC and LRI. Since there are only three free parameters being fit to the cutout from the LRI (x, y, and magnitude), there is no degeneracy and PyGFit easily recovers the position of the high resolution object in the LRI. It is then a simple matter to measure the median difference between the object positions in the HRC and the LRI and correct the HRC accordingly. This also accounts for any miscentering of the PSF, which is important because if the PSF is not properly centered then the galaxy models will also be miscentered after PSF convolution. The fitting process will naturally account for this offset and so the final object positions will be shifted by the PSF offset. Therefore when PyGFit performs the alignment step, it automatically corrects the HRC in such a way that the PSF-convolved galaxy models will be properly aligned with the LRI.

2.4. *Fitting*

2.4.1. *Cutouts*

The first step in the fitting process is to identify all low resolution sources which have matching objects from the aligned HRC. Objects from the HRC are matched with a low resolution source if the object falls on one of the pixels identified by Source Extractor as belonging to the segmentation region. Low resolution sources without any overlapping objects are ignored. Fitting is done with the background-subtracted LRI, and fitting proceeds from the brightest low resolution sources to the faintest. After each source is fit the best fitting model is subtracted from the LRI to remove its contribution to any nearby sources.

PyGFit generates a cutout of the blend from the LRI and extracts a matching cutout from the RMS map. The extracted cutout is square and is large enough to enclose the full segmentation region of the low resolution source. The cutout is further extended in every direction by the size of the allowed position shift during the fitting process, and an extra two pixels of buffer are added on each side. If the resultant cutout image extends off of the LRI then PyGFit shifts the cutout to abut the edge of the image.

2.4.2. *Model Generation*

PyGFit then generates a model image for every matching object from the HRC. Currently PyGFit supports two models: point and Sérsic models. Model generation is very straightforward for point sources, which are simply a copy of the PSF image shifted to match the HRC position and scaled to match the total flux of the first guess used during the fit (Section 2.4.3). Shifting is accomplished with third-order spline interpolation.

Sérsic model generation begins by calculating the average surface brightness (Σ) of the Sérsic model in each pixel of the cutout. The Sérsic profile depends upon the effective radius (r_e), Sérsic index (n), axis ratio (q), position angle (P.A.), total flux (F_{tot}), a boxiness parameter (c), and the profile center (x_{cent}, y_{cent}). From these eight parameters PyGFit derives two more parameters: the surface brightness at the effective radius (Σ_e) and a coupling factor (κ) that ensures that the effective radius is also the half-light radius. The surface brightness as a function of radius is then given by:

$$\Sigma(r) = \Sigma_e e^{-\kappa[(r/r_e)^{1/n}-1]} \tag{1}$$

Where:

$$\Sigma_e = \text{flux} * \mathrm{R}(c)/[2\pi q r_e^2 e^{\kappa} \kappa^{-2n} \Gamma(2n) n] \tag{2}$$

During model generation PyGFit sets the flux according to its first guess for χ^2 minimization (Section 2.4.3). $\Gamma(2n)$ is the gamma function and $\mathrm{R}(c)$ is given by:

$$\mathrm{R}(c) = \frac{\pi c}{4\beta(1/c, 1+1/c)} \tag{3}$$

In this equation β is the beta function with two parameters. All of these definitions precisely match those for

GALFIT (Peng et al. 2002, 2010), which is done intentionally for ease of use. If GALFIT is used to fit Sérsic profiles to the HRI, then the output from GALFIT can be passed directly into PyGFit without modification.

PyGFit must calculate the flux in each pixel of the model image. The most straightforward way to do this is to integrate the Sérsic function over each pixel. However, the integration time of the Sérsic function can be computationally prohibitive, and PyGFit would be dramatically slower if it attempted to integrate the Sérsic function over every pixel. Instead PyGFit performs a numeric integration by splitting each pixel into subpixels, evaluating the Sérsic function at each subpixel, and averaging their values together. The level of resampling is finer towards the center of the model, with different levels of resampling for $r > 2r_e$, $r < 2r_e$, and the central pixel. For these regions PyGFit resamples the model image such that the size of each subpixel is at most $r_e/2$, $r_e/20$, and $r_e/200$, respectively.

Extensive testing has shown that this methodology provides a reasonable execution time without compromising the results. The only exception is for galaxies with small radii and high Sérsic indexes ($n \sim 8$), where we find that the only way to reliably calculate the flux at the center of the Sérsic profile is by directly calculating its integral. However, these cases are easy to detect and, if desired, PyGFit can automatically switch from its default treatment to a full integration to guarantee that all galaxies are properly modeled. After generating the Sérsic model PyGFit then convolves it with the low-resolution PSF.

At the end of the model generation process PyGFit has a model image for every high-resolution object associated with a given low resolution source. The generated model image matches the cutout for the blend. The total flux of the model has been normalized to match the first guess that goes into the χ^2 minimization (Section 2.4.3), and the model has been convolved with the low-resolution PSF. Therefore, all necessary steps have been performed to prepare the model images for fitting to the cutout of the low resolution source.

2.4.3. *Fitting*

PyGFit uses a Levenberg−Marquardt algorithm to minimize χ^2 and fit the models to the low-resolution cutouts. The cutout from the RMS image gives the uncertainty of every pixel in the cutout. Each model has only three free parameters: x, y, and flux. For Sérsic models all other parameters (n, r_e, q, B/A, and P.A.) are held fixed to the values found in the HRC. Since only the position and flux of the objects are free parameters, PyGFit does not need to generate a new model image at every iteration of the χ^2 minimization. Instead, PyGFit takes the stored model image, shifts it to match the new position (using third-order spline interpolation), and rescales it to match the new flux. This is done for point source models as well as Sérsic models. For each iteration of the χ^2 minimization PyGFit takes the adjusted models, adds them together to make a total model image, and then calculates χ^2 in the standard way.

The total number of free parameters (n_f) for each blend is given by $n_f = 3 * n_{HRC}$ where n_{HRC} is the number of objects from the HRC associated with the low resolution source, and the total degrees of freedom for each fit is the number of pixels in the cutout minus the number of free parameters. As a first guess for model positions PyGFit uses the object positions from the HRC after the alignment step. The first guess value for the flux of each model is the magnitude of the object from the HRC converted to a flux using the zeropoint of the LRI.

During the fit the positions are constrained to move within a fixed distance of the first guess. The size of the allowed position shift is easily configurable. Rather than placing a constraint directly on the χ^2 minimization, PyGFit uses a mapping function to convert the position offset calculated by the χ^2 minimization from an infinite range to a finite range. This keeps the positions within the desired offset without any modifications to the χ^2 fitting routine. We also force the fits to have positive fluxes.

2.5. *Final Catalog Output*

After fitting has completed for all low resolution sources PyGFit generates a final catalog. The final catalog combines data from a number of sources. It includes the best fitting magnitudes and fluxes for all matching objects in the HRC. It includes the object number and auto-magnitude for the low resolution source from Source Extractor, plus any other requested Source Extractor parameters. All the information from the HRC is copied to the final output catalog. Finally, PyGFit computes a number of diagnostic measures which can be included in the output catalog. These include values such as the total number of high-resolution objects associated with the low-resolution source, the distance and fitted magnitude of the nearest object in the blend, the fitted magnitude of the brightest object in the blend, the total fitted flux and magnitude of all objects in the blend, and the fraction of the blend flux which is accounted for by each object.

3. APPLYING PYGFIT TO REAL DATA

Our initial test case for PyGFit involved measuring SEDs of galaxies in high redshift galaxy clusters. The data and project are described in detail in Mancone et al. (in preparation). In summary, we use 13 galaxy clusters with $1 < z < 1.9$ observed with broadband photometry in eleven filters. All the clusters were observed with ground-based optical imaging in the Bw, R, and I bands, ground-based NIR imaging in the J, H, and Ks bands, space-based NIR imaging in all four IRAC bands, and finally *HST* WFC3/F160W imaging. We use GALAPAGOS (Häußler et al. 2011) to run GALFIT and fit a single Sérsic profile to every galaxy in our F160W images. We then run PyGFit on each of the bands using the GALFIT catalog as the HRC.

Figure 2 illustrates typical results from PyGFit. It shows the original images in four different bands and their residuals after fitting in the center of ISCS J1434.5+3427, a galaxy cluster at $z = 1.243$. PyGFit cleanly subtracts all the visible objects. This is very typical for our ground-based imaging, especially in the NIR where the residuals are indistinguishable from sky noise in virtually all cases.

At 4.5μm (far right of Figure 2) there are small residuals visible in the very centers of many objects. These residuals are common to both our 3.6μm and 4.5μm filters but are not visible in the 5.8 and 8.0μm filters. The

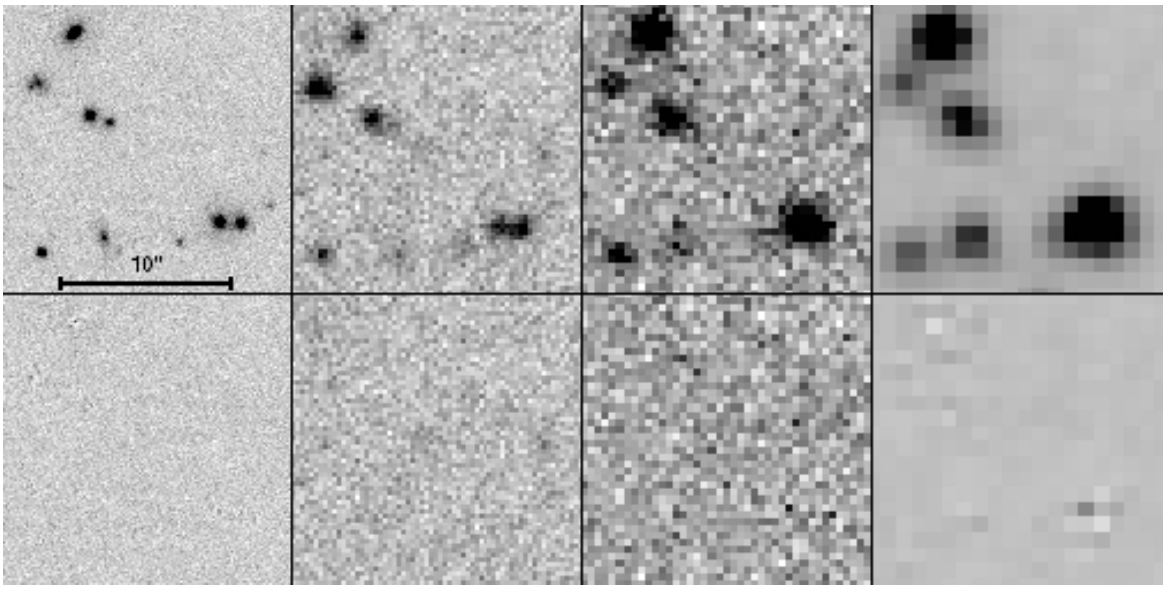

FIG. 2.— Original images (top row) and residuals (bottom row) from our GALFIT and PyGFit runs in the core of a high redshift ($z = 1.243$) galaxy cluster. From left to right the images correspond to WFC3/F160W, R, H, and 4.5μm. The WFC3/F160W image was fit with GALFIT, while all other bands were fit with PyGFit. All panels show the same field of view, and the scale in the top left panel is $10''$ long.

primary difference between the IRAC images is the size of the PSF, which varies from $1.66''$ to $1.98''$ (Fazio et al. 2004). Our IRAC images have been dithered and resampled to have a pixel scale of $0.865''$/pixel, which means that the PSF is properly sampled only if it is larger than $1.73''$. The 4.5μm imaging falls just short of this requirement with a mean PSF size of $1.72''$, while the 5.8μm imaging has a mean PSF size of $1.88''$. Therefore, the PSF is resolved for our 5.8 and 8.0μm images but is suboptimally sampled for our 3.6 and 4.5μm imaging. Without a fully resolved PSF interpolation (which PyGFit does during model generation and fitting) can introduce artifacts, and this is likely the source of the small residuals observed in our blue IRAC bands. However, our simulations (Section 4) conclusively demonstrate that PyGFit can reliably extract magnitudes and fluxes from the observations, and that the primary source of uncertainty is simply sky noise.

An examination of our residuals images reveals a few classes of problems which can result in PyGFit failures. We show a few examples of these cases in Figure 3. One source of difficulty is when a galaxy is not well represented by a Sérsic function. In the example in Figure 3 (far left) a galaxy has extended features which cannot be modeled by a single Sérsic profile. As a result, the central region of the galaxy is over-subtracted, while the outer region is under-subtracted. As long as the galaxy does not have a substantial amount of flux outside of the model radius, PyGFit can still return an approximately correct total flux. If the galaxy does have substantial flux outside of the model radius, PyGFit will underestimate the total flux of the galaxy. However, any error introduced by a mismatched model will be the same for all filters. Therefore, when using PyGFit to measure SEDs, this class of problem can lead to an underestimated SED normalization but will not introduce any additional filter-to-filter uncertainty in the SED.

PyGFit can fail catastrophically when objects in the LRI are missing from the HRC. If, in the LRI, an object in the HRC is blended with another object which is not in the HRC, then PyGFit will assign flux from the second object to the first, overestimating its flux. This can happen in a number of ways, two of which are illustrated in Figure 3. The top central panel of Figure 3 shows a faint galaxy. The center panel shows that, in the 4.5μm image, there appears to be a significant elongation towards the bottom right, which cannot be accounted for from the F160W image. After subtraction (bottom center) there appears to be an object left over below and to the right of the F160W source. The only way to explain this is with the presence of an object which is bright in 4.5μm but nearly invisible in F160W, and which happens to be blended with the object visible in F160W. As a result, the object from the HRC is overfit to account for the flux from the additional low-resolution object, and therefore its fit is unreliable.

The right set of panels in Figure 3 show the same class of problem in another context. This shows what can happen when the LRI extends past the HRC. The top right panel shows an object which is near the edge of the F160W image. In the 3.6μm image (center right) a bright object happens to be nearby but is just outside of the F160W field of view, and is therefore missing from the HRC. Although this second object is outside of the F160W field of view, it is bright enough to contribute substantially to the flux near the object of interest. As a result, PyGFit overestimates the 3.6μm flux of the object which is in the HRC. While this particular problem can likely occur for any image, we see it most commonly in our IRAC images. This is because our IRAC images have the highest source densities and the largest PSF of all of our images, and this combination increases the likelihood of having such a blend.

Obviously, PyGFit cannot account for objects which are missing from the HRC. This fact should be kept in mind when using PyGFit and care must be taken to include all sources which will be visible in the LRI, or to reject sources that are blended with objects missing from

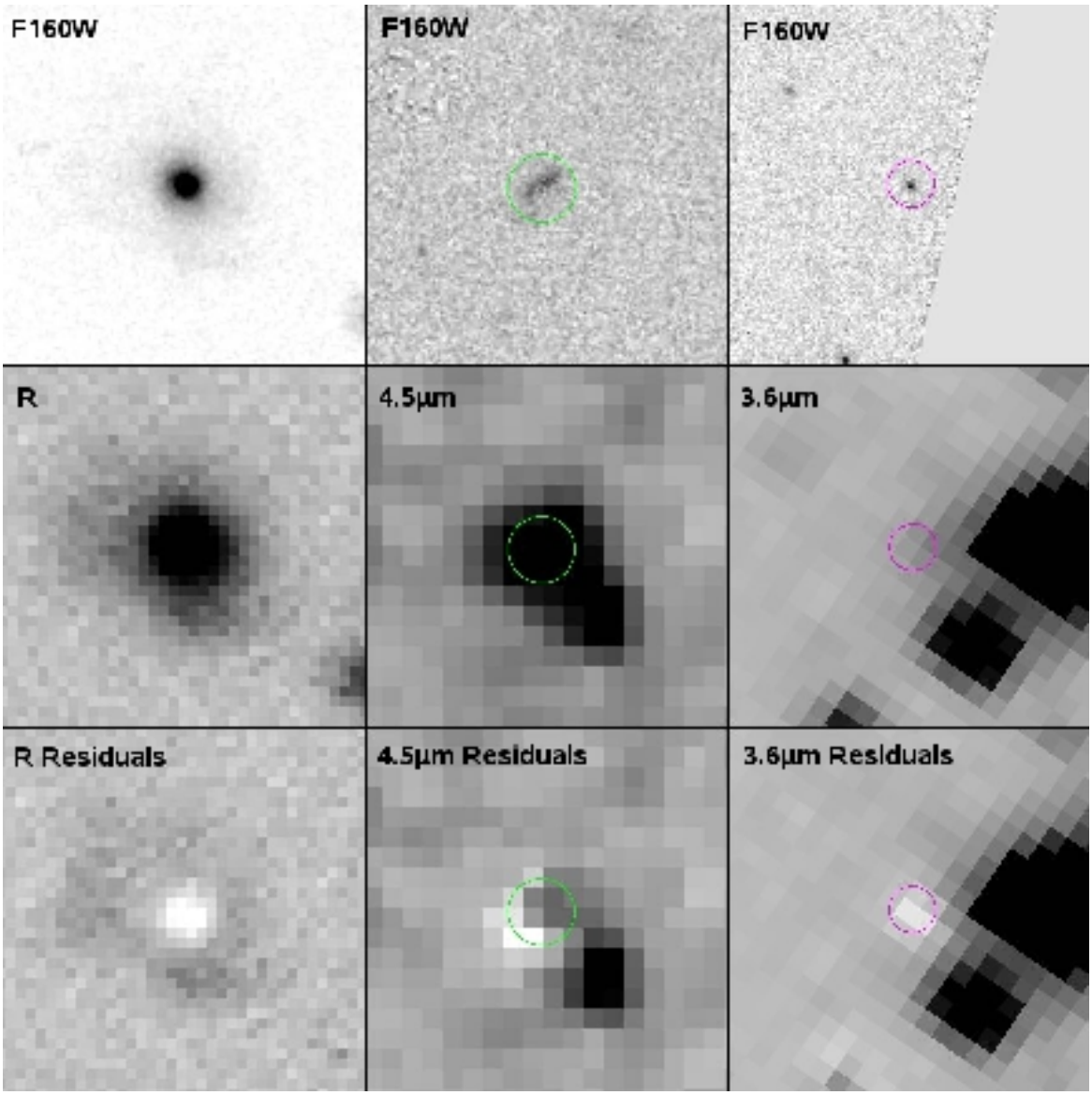

FIG. 3.— Three examples of cases where PyGFit can fail. The top row of panels shows the F160W images used to create the HRC. The center row of panels shows the LRI with the same field of view, while the bottom row of panels shows the residuals of the LRI after fitting. The LRI for the left uses ground-based R band imaging, while the example in the center is taken from the 4.5μm imaging and the example on the right is taken from the IRAC 3.6μm imaging. The left column shows a galaxy with extended features which cannot be described by a Sérsic profile. The center column shows a galaxy which is isolated in F160W but which is blended with another source in 4.5μm. The right column shows a galaxy near the edge of the F160W image which is blended with a bright source which is outside of the F160W image. Further details are in the text.

the HRC.

4. SIMULATIONS

We use simulations to estimate the errors for the fitted magnitudes and fluxes from PyGFit, as well as to evaluate its fidelity and limitations. To aid in this process we developed a companion routine to PyGFit named PyGSTI (Python Galaxy Simulation Tool for Images). PyGSTI uses the same model generation routines developed for PyGFit, generates simulated galaxies, and inserts them into images. We have designed PyGFit to use PyGSTI in a fully automated fashion. We note that while PyGSTI is packaged with PyGFit, it can also run as a stand-alone program and is convenient for generating simulated galaxies and stars for any number of applications.

When running simulations PyGFit randomly selects galaxies from the HRC, assigns them magnitudes from the luminosity function of the LRC, places them into random locations in the original image, runs PyGFit on the simulated frame, and repeats this process many times. PyGFit limits the number of artificial galaxies placed into each simulated frame to prevent an excessive increase of crowding. By default, the source density in PyGFit's simulated frames is 2.5% higher than in the original LRI, and PyGFit generates 100 simulated frames. Both of these parameters are easily configurable.

Once PyGFit runs on all simulated frames, a final catalog is created with input and output magnitudes and fluxes for the simulated galaxies, along with all of the output parameters normally recorded by PyGFit (Section 2.5). This includes information on the number of objects

the simulated galaxy was blended with, how close and bright the nearest neighbor is, and other environmental indicators.

4.1. *Simulation Results*

The first result that our simulations show is that for a small percentage ($\sim 2\%$) of galaxies PyGFit dramatically underfits the model, effectively assigning them zero flux. This only happens for faint galaxies in blends, and happens for all our filters. These galaxies are easily detected in both the simulated catalogs and the real catalogs. We find that simulated galaxies with a fitted magnitude more than five magnitudes fainter than the brightest galaxy in the blend have almost always been affected by this problem. We find similar galaxies in our real data and note that they are easily detected by the same criteria. Such galaxies should always be removed from any real sample as their magnitudes are completely unreliable. Similarly, we remove them from our simulated galaxy sample and exclude them from further analysis.

We show the results of our simulations for three filters (R, J, and 3.6μm) in Figure 4. The top row of panels in this Figure shows the input and output magnitude of each simulated galaxy. The bottom row of panels shows the corresponding error as a function of magnitude which is calculated by binning the simulated galaxies in magnitude space and measuring the standard deviation in each bin. Error bars are calculated with bootstrap resampling. The solid curve in the bottom row of panels shows an estimate of the magnitude error introduced by sky noise for an aperture magnitude with a diameter of $4''$. We also show the error that results from using a $4''$ diameter aperture-magnitude.

For the R and J bands there is excellent agreement between the magnitude errors that result from PyGFit and the error that results from sky noise for a $4''$ diameter aperture-magnitude. While the magnitudes returned by PyGFit are not an aperture magnitude (they are instead a model fit), this still provides a good reference point for comparison. The fact that the magnitude errors from PyGFit are very similar to the plotted sky-noise limit suggest that for the R and J bands PyGFit's performance is primarily limited by sky noise, which sets a fundamental limit for any method that measures the flux of an object. We note that PyGFit does substantially better than a simple $4''$ aperture magnitude (open circles in Figure 4). This is not surprising, as aperture magnitudes are not robust in crowded environments.

However, PyGFit does not reach the sky-noise limit for a $4''$ aperture magnitude in our 3.6μm data. A close examination of the top right panel of Figure 4 shows that there are poorly fit galaxies ($|\text{Mag In} - \text{Mag Out}| > 0.75$) driving this scatter. Our simulations reveal that PyGFit begins to break down when two galaxies are very close together or when a galaxy is blended with a much brighter one. To show this we perform another simulation where we insert pairs of galaxies into an image with the same noise properties, pixel scale, and PSF as the 3.6μm image. These simulated pairs have separations between $0.2''$ and $3''$, magnitude differences between 0 and 3 (i.e. flux ratios between 1 and ~ 15), and the brighter galaxy in the pair has a magnitude between 15 and 17. We drop these pairs into an otherwise blank image and measure PyGFit's fidelity as a function of flux ratio and separation for close pairs.

Figure 5 illustrates the result. The left panel shows the magnitude error for simulated galaxies as a function of the flux ratio of a galaxy and its neighbors. To isolate the influence of the flux ratio, this panel excludes galaxies separated by less than the FWHM of the PSF ($1.66''$ in 3.6μm). The right panel shows the magnitude error for simulated galaxies as a function of the separation between the pair relative to the FWHM of the PSF. This panel only shows galaxies which are the brightest galaxy in the pair. We find that our 3.6μm PyGFit results become unreliable for galaxies with flux ratios < 0.25 or separations $\lesssim 60\%$ ($\lesssim 1''$) of the PSF radius. Tests show that our other filters encounter a similar issue for such pairs. However, source density is by far the highest in our IRAC images. Because the crowding is less of an issue in our other bands, these limits have a smaller impact in our real and simulated data for our non-IRAC bands. We therefore remove these simulated galaxies from our sample and plot in Figure 6 the fidelity of PyGFit's results for the remaining galaxies.

Figure 6 shows that after removing this problematic case of galaxies from our sample, the quality of the IRAC results is much better. We note that no algorithm can deblend objects which are arbitrarily close together, or which have been blended with an arbitrarily brighter object. Indeed, a close pair is only considered resolved when its members are separated by at least one PSF FWHM. However, PyGFit is reliably fitting galaxies separated by $\sim 60\%$ of the PSF radius, which shows that it is a viable option for crowded fields.

We examine how PyGFit performs as a function of environmental diagnostics. In Figure 7 we show the fidelity of PyGFit's results as a function of the number of objects blended together, the distance to the nearest object, the flux ratio between an object and its nearest neighbor, and the fraction of the blend accounted for by the simulated object. We only plot simulated galaxies in this figure if they have $[3.6] < 19.0$ and pass the cuts discussed above (i.e. flux ratio > 0.25 and separation $> 1''$). There are no strong correlations in Figure 7, demonstrating that the quality of PyGFit's results are independent of the degree of crowding or other environmental factors. Similarly, we also find that there is no relationship between the uncertainty of PyGFit's results and any of the Sérsic parameters. Other than magnitude, PyGFit's fidelity is independent of r_e, n, B/A, and position angle.

5. CONCLUSIONS

We present PyGFit, a program which generates PSF-matched photometry from images with disparate pixel scales and PSF sizes. PyGFit takes model fits from high resolution images, fixes shape parameters, and fits the models to low resolution images allowing only the magnitudes to vary along with small position shifts. We apply PyGFit to real images and also perform simulations to measure PyGFit's fidelity. With the exception of some small residuals in the two bluest IRAC filters where the PSF is undersampled, PyGFit is able to cleanly subtract galaxies from the LRI. Especially in the ground-based images, where the PSF is well resolved, there appears to be nothing left in the residual images but sky noise. Simulations show that the uncertainty in PyGFit's magnitudes are consistent with being limited by sky noise.

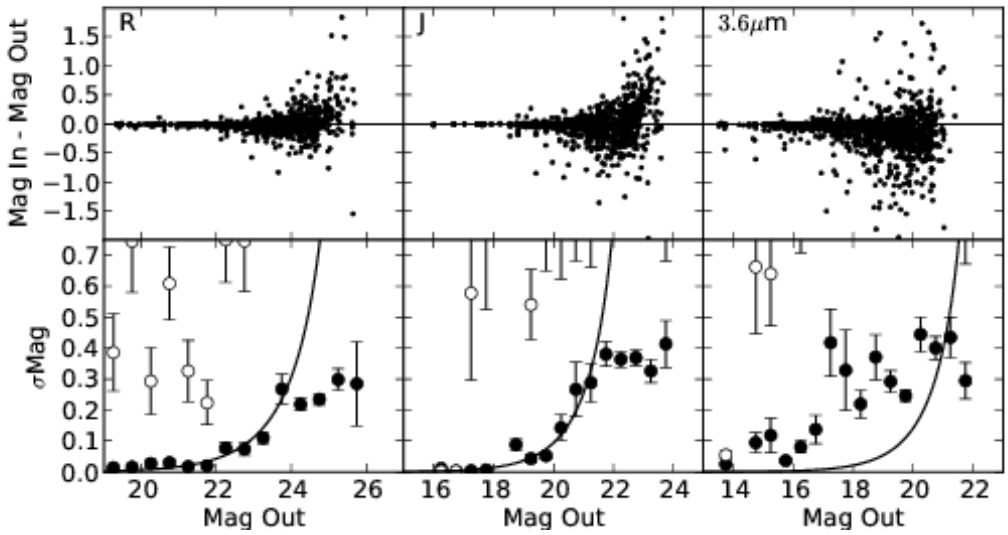

FIG. 4.— (Top) Input magnitude versus input magnitude minus output magnitude as measured by PyGFit simulations for three filters: R (left), J (center), and 3.6μm (right). (Bottom) Magnitude error as a function of magnitude. The solid circles show the error of PyGFit's magnitude estimates in magnitude bins. For comparison the open circles show the resulting error when a 4″ diameter aperture-magnitude is used instead of PyGFit. The solid line shows the uncertainty introduced by sky noise for an aperture with a 4″ diameter.

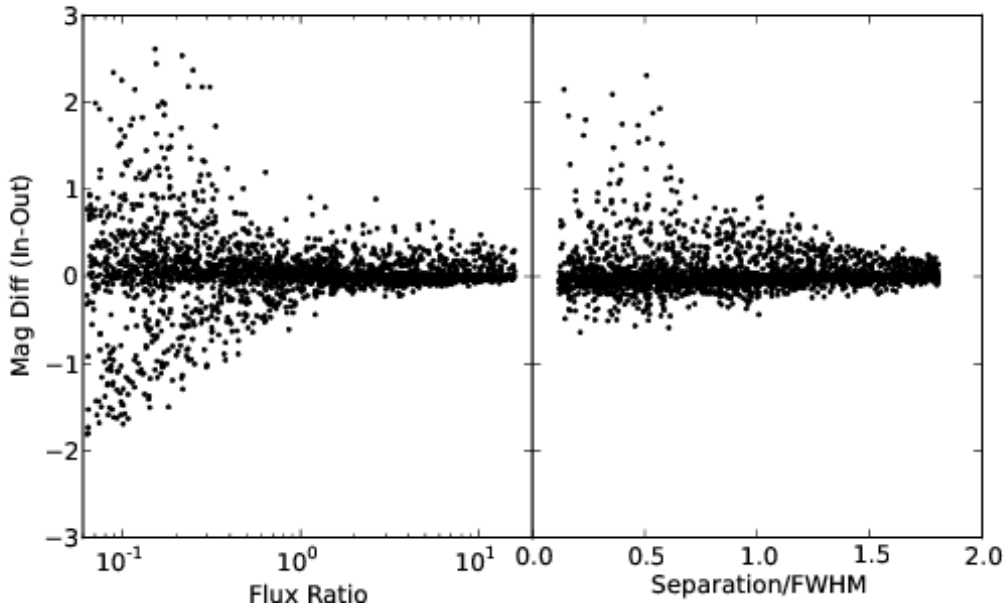

FIG. 5.— Difference between input and output magnitude for simulated galaxies as a function of flux ratio (left) and separation relative to the size of the PSF (right). To cleanly separate the two competing effects the left panel only includes galaxies separated by at least 1 PSF FWHM, and the right panel only includes galaxies which are the brightest galaxy in the pair.

Our simulations identify a few classes of problems which can introduce errors into the PyGFit results. Most important are catalog problems, i.e. incorrect or missing high resolution data. Primary examples of catalog problems include fitting galaxy models in the HRI which are a poor fit to the galaxy, or the presence of objects in the LRI that are missing from the HRC. The latter commonly happens because of differences in filter wavelength or because of the finite size of the HRI.

Our simulations show that a small fraction ($\sim 2\%$) of faint, blended galaxies are fitted away and have effectively been assigned zero flux. While we find no obvious predictors for when this happens, such cases are rare and easy to detect/remove. We further find that (as expected) PyGFit cannot deblend galaxies with arbitrarily close neighbors or arbitrarily bright companions. This effect is important for our 3.6μm data where crowding is the most prominent. We find that PyGFit's results are reliable down to separations as small as $\sim 60\%$ of the PSF size.

REFERENCES

Bertin, E. & Arnouts, S. 1996, A&AS, 117, 393

de Santis, C., Grazian, A., Fontana, A., & Santini, P. 2007, New Astronomy, 12, 271

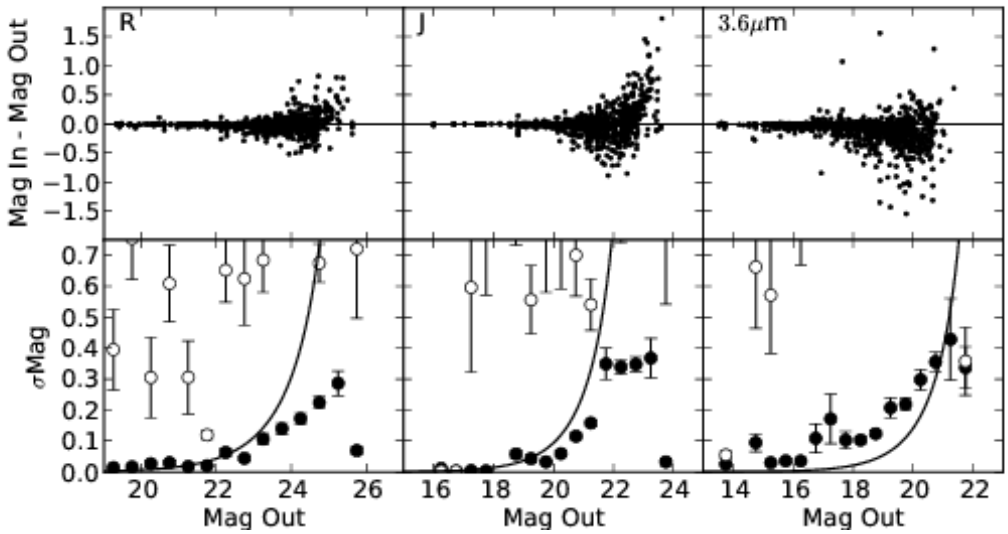

FIG. 6.— Same as Figure 4, except simulated galaxies with separations < 60% of the PSF radius or flux ratios < 0.25 have been cut from the sample.

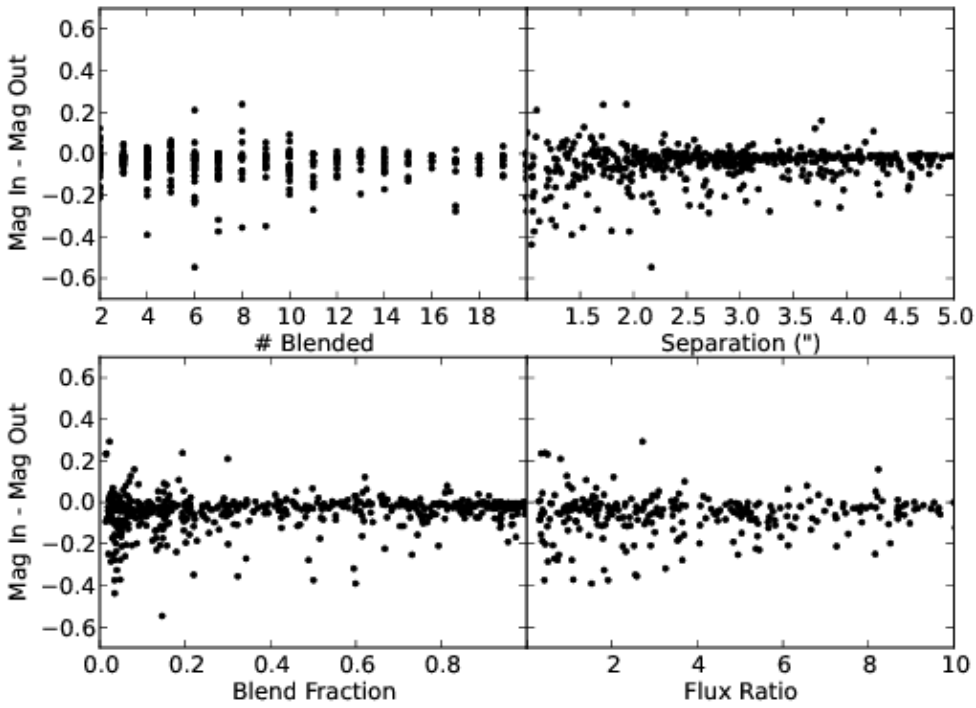

FIG. 7.— PyGFit errors as measured with our simulations for our 3.6μm galaxies with [3.6] < 19.0 versus the number of objects in the blend (top left), the distance to the nearest blended object (top right), the flux ratio between the object and its nearest neighbor (bottom right), and the fraction of the blend flux accounted for by the simulated object (bottom left).

Dressel, L. 2011, Wide Field Camera 3 Instrument Handbook, Version 4.0 (Baltimore, MD: STScI)

Fazio, G. G., Hora, J. L., Allen, L. E., Ashby, M. L. N., Barmby, P., Deutsch, L. K., Huang, J.-S., Kleiner, S., Marengo, M., Megeath, S. T., Melnick, G. J., Pahre, M. A., Patten, B. M., Polizotti, J., Smith, H. A., Taylor, R. S., Wang, Z., Willner, S. P., Hoffmann, W. F., Pipher, J. L., Forrest, W. J., McMurty, C. W., McCreight, C. R., McKelvey, M. E., McMurray, R. E., Koch, D. G., Moseley, S. H., Arendt, R. G., Mentzell, J. E., Marx, C. T., Losch, P., Mayman, P., Eichhorn, W., Krebs, D., Jhabvala, M., Gezari, D. Y., Fixsen, D. J., Flores, J., Shakoorzadeh, K., Jungo, R., Hakun, C., Workman, L., Karpati, G., Kichak, R., Whitley, R., Mann, S., Tollestrup, E. V., Eisenhardt, P., Stern, D., Gorjian, V., Bhattacharya, B., Carey, S., Nelson, B. O., Glaccum, W. J., Lacy, M., Lowrance, P. J., Laine, S., Reach, W. T., Stauffer, J. A., Surace, J. A., Wilson, G., Wright, E. L., Hoffman, A., Domingo, G., & Cohen, M. 2004, ApJS, 154, 10

Fernández-Soto, A., Lanzetta, K. M., & Yahil, A. 1999, ApJ, 513, 34
Häußler, B., Barden, M., Bamford, S. P., & Rojas, A. 2011, in Astronomical Society of the Pacific Conference Series, Vol. 442, Astronomical Data Analysis Software and Systems XX, ed. I. N. Evans, A. Accomazzi, D. J. Mink, & A. H. Rots, 155
Komatsu, E., Smith, K. M., Dunkley, J., Bennett, C. L., Gold, B., Hinshaw, G., Jarosik, N., Larson, D., Nolta, M. R., Page, L., Spergel, D. N., Halpern, M., Hill, R. S., Kogut, A., Limon, M., Meyer, S. S., Odegard, N., Tucker, G. S., Weiland, J. L., Wollack, E., & Wright, E. L. 2011, ApJS, 192, 18
Labbé, I., Huang, J., Franx, M., Rudnick, G., Barmby, P., Daddi, E., van Dokkum, P. G., Fazio, G. G., Schreiber, N. M. F., Moorwood, A. F. M., Rix, H.-W., Röttgering, H., Trujillo, I., & van der Werf, P. 2005, ApJ, 624, L81
Laidler, V. G., Papovich, C., Grogin, N. A., Idzi, R., Dickinson, M., Ferguson, H. C., Hilbert, B., Clubb, K., & Ravindranath, S. 2007, PASP, 119, 1325
Makovoz, D. & Marleau, F. R. 2005, PASP, 117, 1113
Mancone et al., C. in preparation, ApJ, 701, 428
Peng, C. Y., Ho, L. C., Impey, C. D., & Rix, H.-W. 2002, AJ, 124, 266
—. 2010, AJ, 139, 2097