GIM2D[1] Cookbook

Kim-Vy Tran

Revised: September 19, 2000

1 Introduction

This cookbook serves as a bare-bones explanation of how to use most of the tasks in GIM2D. It should be used with the basic GIM2D pipeline and notes from the GIM2D 2000 Workshop; both are available on the GIM2D webpage. The cookbook is compiled from my experiences with all the different routines explained here, and I encourage you to experiment beyond these examples. I intend to expand this cookbook as soon as I try the other routines not covered here. If you have suggestions or nifty tricks, please let me know (vy@ucolick.org) so I can include them.

As this is not an official publication, it cannot be referenced formally. I would appreciate it, however, if you mention this work in the acknowledgements if you use this cookbook.

2 Getting Started

2.1 Images

2.1.1 HST WFPC or NICMOS Images

Combine multiple exposures in the usual way. If the data are dithered, **DO NOT** use the drizzle task in IRAF.STSDAS.ANALYSIS to combine them. Drizzling the images will keep the correct world coordinates (α, δ) and apply any pixel corrections but its technique of putting data in "drops" does not propagate the noise statistics correctly. Dithered images should not be regridded either as running GIMFIT on regridded galaxy images recovers the regridding pattern. Instead, use MGIMFIT2D (see §3.2) to fit the multiple images of the same object simultaneously.

Only work with the single WF chip images. Do not mosaic the images for the following reasons:

- The chips have slightly different gains and pixel scales. This information is lost when they are mosaicked. Also, the pixels are no longer "square" and the noise statistics are not propagated correctly.
- Mosaicking images creates noisy seams at the chip edges.

[1]Galaxy surface brightness fitting software written by Luc Simard (Simard *et al.* 2000, 1999; Tran *et al.* 2000; Marleau & Simard 1998)

- Mosaicking the pointing will *rotate* the chips, making the chip coordinates different from the mosaic coordinates. The correct chip coordinates are crucial in generating a PSF with TinyTim.

2.1.2 Ground-Based Images

Images should be reduced in the usual way. If multiple images are combined, note if they are averaged or summed. In GIMFIT, the parameter $nframe$ is 1 if the images are summed but n if n frames are averaged. This is important for propagating the noise statistics correctly.

2.2 SExtractor

GIM2D needs to know which pixels belong to the galaxy being fitted as well as which pixels are sky. SExtractor (Bertin & Arnouts 1996) does an excellent job of determining a galaxy's isophotal area, even with near neighbors. Run SExtractor to obtain the catalog files listing (x,y) position, local sky background, local sigma, magnitude, etc., and to output a segmentation image. The segmentation image has the same integer value (value> 0) for all the pixels in the galaxy's isophotal area, assigns sky pixels (value=0), and flags bad pixels (value=-2). The first two are used to determine the models while the last one is also important as it excludes spurious information from the fits.

If there are regions in the image that you do not want to use, assign these pixels a value=-2 to exclude them from the modeling.

2.3 PSFs

2.3.1 TinyTim PSFs

As the PSF varies across the chip, it is useful to make a PSF for each galaxy with TinyTim. In this case, the correct chip coordinates are important; these are taken from the .hhh stacked WFPC images.

Oversampling the PSF by 5 is recommended because WF chips are undersampled. For this oversampling, a PSF of 2.4" in diameter is best. If oversampled PSFs are used, note that GIM2D will rebin and apply the correction kernel (see TinyTim manual) so it is unnecessary for the user to do so. Also, be sure to change the *osamp* and $psftype$ parameters in GIMFIT.

As it takes quite a long time to generate PSFs with TinyTim, creating a PSF for each object in a large sample is inconvenient. What I do is create a PSF star every 50 pixels across the chip (256 stars for each chip). Then I find the closest PSF star to each object using a short program. Whether you are using HST or ground-based images, always be sure the chip is well-sampled with PSF stars.

If the output models have a residual box, increase the PSF size. If the galaxies are much larger than the PSF, a ghost image of the PSF box appears in the output models and residual images.

2.3.2 DAOPHOT PSFs

For ground-based images, the best way to construct PSFs is with DAOPHOT. Creating PSFs with DAOPHOT is a bit more complicated than with TinyTim but is easy enough with scripts. After getting the DAOPHOT software, use the script *doall*494 that Mike Bolte and Katy Flint wrote to get the PSF. This script needs a list of PSF stars from the image in a file; the file is created using IRAF's imexam task with the 'a' key on the stars. Run DAOPHOT once to model the PSF across the chip, then again to create fake PSF stars for each object (you will need a blank image in which the the fake stars are imbedded). Make sure the PSFs do not have negative values.

3 GIMFIT, MGIMFIT, and C2GIMFIT

3.1 GIMFIT

This is the most basic of the three fitting routines. By running GIMFIT in each filter separately, galaxy color gradients can be measured.

1. Combine common images in the usual manner. If images are averaged or medianed, note the number of images. This is important in GIM2D: in GIMFIT the parameter $nframe$ is the number of *averaged* images. If images are summed, $nframe = 1$.

2. Copy individual chips from .hhh file.

3. Run SExtractor on the clipped images. Among the output params in the SExtractor catalog must be pixel (x,y), best magnitude and error, isophotal area, and background sigma. Optional params I include are RA & Dec, and star/galaxy classifier. Also output a segmentation image.

4. Create PSF stars for each galaxy. In TinyTim, the (B-V) color of the PSFs does not affect GIM2D's modeling. Rename output PSFs to gx*y*_p.fits.

5. Using gawk, create a .gfxt list for each image to use in GSCRIPT. The list must have x, y, background sigma, junk (I use the magnitude here), and isophotal area in this order.

6. Edit parameters in GIMFIT. This ensures that the cl script created by GSCRIPT has the correct parameter values. Example GIMFIT parameters:

```
in_image = " "              Name of input image
out_image = ""              Name of output image
(sym_image = "")            Name of symmetrized image
(res_image = "")            Name of residual image
(msk_image = "")            Name of pixel mask image
(logfile = "")              Name of log file
(mdparfile = "deV.mdpar")   Name of parameter value file
(initparam = yes)           Determine parameters from input image moments
(dosym = no)                Symmetrize input image (yes/no)?
(dobkg = yes)               Recompute background parameters (yes/no)?
(imscale = 0.07)            Image scale (arcsecond/pixel)
(psftype = "tiny_wfpc")     PSF type (delta|gaussian|tiny_wfpc|tiny_nic|tinytim|user)
(in_psf = "")               Name of tinytim/user PSF image
(seeing = 0.07)             Seeing FWHM (arcseconds)
(bkgsig = )                 Background sigma (ADU)
(nsamp = 200)               Number of parameter space samples
(ccdgain = 7.)              CCD Gain (electrons/DU)
(nframe = 3)                Number of frames combined
(metseed = -24781)          Metropolis seed
(osamp = 2)                 Core oversampling factor
(mode = "q")
```

dobkg : Lets GIMFIT recompute background values. Always set to yes.

psftype : If set to "tiny_wfpc" or "tiny_nic", GIMFIT rebins PSF by *osamp* factor to pixel scale and applies the correction kernel. For "delta", GIMFIT won't use a psf-convolved model; very useful for large galaxies that are well-sampled anyway because it reduces running time significantly.

The parameters *imscale* & *seeing* are used only when $psfytpe = gaussian$.

nsamp : The higher the value, the more rigorous GIMFIT will be about testing a model. Larger values increase running time. Usual range is 200-400.

$nframe$: Number of frames averaged. If frames are summed, $nframe = 1$.

metseed : Always a large negative odd integer between 0 and -32000.

osamp : How much the TinyTim PSF pixels are oversampled compared the the galaxy pixels. Used for WFPC imaging since TinyTim can create oversampled PSFs.

$mdparfile$: File with parameter ranges. GIMFIT will not run without it. For a deVaucouleurs plus exponential disk model, the mdpar file has:

```
300                              number of initial models
500.0 500.0 1.0 1000.0 1000.0    total flux
0.5 0.5 0.0 1.0 1.0              bulge fraction (B/T)
2.5 2.5 0.0 5.0 5.0              bulge effective radius
0.35 0.35 0.0 0.7 0.7            bulge ellipticity
90.0 90.0 -360.0 360.0 180.0     bulge position angle
2.5 2.5 0.0 5.0 5.0              exponential disk scale length
45.0 45.0 0.0 85.0 85.0          disk inclination
90.0 90.0 -360.0 360.0 180.0     disk position angle
0.0 0.0 -1.0 1.0 0.1             subpixel x offset of galaxy center
0.0 0.0 -1.0 1.0 0.1             subpixel y offset of galaxy center
0.0 0.0 0.0 0.0 0.0              background level
4.0 4.0 4.0 4.0 0.0              exponential power index n (as in r^(1/n))
```

In the last 11 parameters, the first value is the initial guess. The second is the best value for the parameter. In the mdpar file, these two numbers should be the same; of course, the best value changes when GIMFIT models the galaxy. The third and fourth value are the min and maxes for the parameter, and the fifth the initial perturbation.

The only parameters I change in this file are the initial number of models (anywhere from 200 to 500), index r, and the bulge fraction.

Different Profiles:

de Vaucouleurs + exponential disk:
r line: 4.0 4.0 4.0 4.0 0.0

Sersic Only:
r line: 2.1 2.1 0.2 4.0 3.8
B/T line: 1.0 1.0 1.0 1.0 0.0

Sersic Bulge + exponential disk:
r line: 2.1 2.1 0.2 4.0 3.8

Exponential + exponential:
r line: 1.0 1.0 1.0 1.0 0.0

7. Edit GSCRIPT and run. Example GSCRIPT parameters:

```
in_image= I.fits          Name of input image
in_file = xgal.in         Name of input SExtracted file
(out_file1= I.xgal)       Name of output XGAL file
(out_file2= I.cl)         Name of output IRAF GIMFIT2D script
(path = /data/Images/)    Directory path
(seed = -31231)           Random generator seed
(rmin = 30.)              Minimum extraction radius
(earea = 10.)             Extraction area
(varpsf = yes)            Are you using a different PSF for each object?
(psfname= )               PSF name to use when varpsf=no
(mode = q)
```

rmin : The minimum radius in pixels of the galaxy thumbnails. The minimum value for this should be $\geq$ radius of the PSF thumbnail. Remember to use the image's pixel scale, so $rmin = 12$ for TinyTim PSF stars 2.4" in diameter, even if they are oversampled by 5. Objects less than *rmin* from chip edge do not get fitted.

$earea \times iso\_area \sim area_{thumbnail}$: The area of the galaxy thumbnail is determined by *earea* and the isophotal area measured by SExtractor. Values less than 10 are not good (not enough sky pixels). Best range is 10-20. If the object is more than *rmin* away from the chip edge but *earea* is goes off the chip edge, xgal still clips a thumbnail with the object centered and tags the area off the chip as bad pixels.

varpsf : Change to no only if using one psf for all galaxies.

8. Run XGAL to extract galaxy thumbnails from original and segmentation images.

9. Run GIMFIT in the background with:

cl < *name.cl* > & *machine.log* &

GIMFIT can run on multiple machines simultaneously to fit galaxies in the same directory.

10. GIMFIT has now created an output (gx*y*_o.fits) and residual image (gx*y*_r.fits), and a logfile (gx*y*.log) for each object. The final values for the 11 input parameters are at the end of the logfile preceded by the letter "P"; they are also put in the gx*y*_r.fits header. The errors listed are 3σ. The χ^2_ν, $r_{1/2}$, asymmetry and symmetry indices, and concentration are also shown. The asymmetry and symmetry indices are calculated every $n \times r_{1/2}$ where $n = 1 - 10$.

3.2 MGIMFIT

Use MGIMFIT if the data have been dithered. It simultaneously fits n images of a given galaxy in a filter. The individual images are weighted by their exposure times.

Steps 1–5 same as for GIMFIT. Be sure to run SExtractor and have a .gfxt file for each dithered pointing.

6. Edit MGIMFIT parameters. This is the same as the GIMFIT parameters (see GIMFIT Step 6) with the addition of:

```
(deepref = no)     Is the reference image a deeper image (yes/no)
(offrecalc = yes)  Will MGIMFIT2D recalculate image offsets (yes/no)
```

Do not change *deepref*; it is not what you think it is.

7. Create a working (sub) directory for all the images and cd to it. Create a GGROUP input file with pathname to science image, image rootname, .gfxt filename, and (x,y) pixel offsets from the primary ("reference") image. For example, file "ggroup1.in" contains:

/data/mgimfit/ F1_I2_c2 F1_I2_c2.gfxt 0.0 0.0
/data/mgimfit/ F1_I1_c2 F1_I1_c2.gfxt 2.5 -3.0

8. Run GGROUP to create the cl script and extract the galaxy thumbnails. Example GGROUP parameters:

```
in_file = "ggroup1.in"              Name of input GGROUPER file
(reduce_scrip = "I.cl")             Name of output IRAF GIMFIT2D script
(path = "/data/mgimfit/deV814/")    Directory path
(seed = -26713)                     Random generator seed
(rmin = 12.)                        Minimum extraction radius
(earea = 10.)                       Extraction area
(varpsf = yes)                      Are you using a different PSF for each object?
(deepref = no)                      Is reference image a deeper image (yes/no) ?
(forcelist = no)                    Impose reference image object list to all image
(offrecalc = yes)                   Will MGIMFIT2D recalculate image offsets (yes/n
(psfname = "")                      PSF name to use when varpsf=no
(sigimage = no)                     Are you using sigma images?
(mode = "q")
```

See GIMFIT Step 7 for explanation of parameters. Do not change *deepref*; it is not what you think it is.

8. Run MGIMFIT in the background with:

cl < *name.cl* > & *machine.log* &

MGIMFIT can run on multiple machines simultaneously to fit galaxies in the same directory.

9. MGIMFIT has now created output (gx*y*_o*.fits) and residual images (gx*y*_r*.fits) for each input image of a given galaxy, and a logfile (gx*y*.log). The final values for the 11 input parameters are at the end of the logfile preceded by the letter "P"; they are also put in the gx*y*r*.fits header. The errors listed are 3σ. The logfile also lists the χ^2_ν, $r_{1/2}$, asymmetry and symmetric indices, and concentration for each image of the given galaxy.

3.3 C2GIMFIT

Use C2GIMFIT to fit simultaneously a galaxy's image in two filters. This is useful for measuring bulge and disk colors. All the structural parameters except for the total flux, B/T, and sky background are the same.

C2GIMFIT works exactly the same way as GIMFIT except it uses object lists as the input:

```
in_list = "@gal.in"        List of input image(s)
out_list = "@gal_o.in"     List of output image(s)
(sym_list = "none")        List of symmetrized image(s)
(res_list = "@gal_r.in")   List of residual image(s)
(msk_list = "@gal_m.in")   List of pixel mask image(s)
(sig_list = "none")        List of sigma image(s)
(logfile = "gal.log")      Name of log file
```

The input files list the thumbnails to be used. Always list the better image first.

C2GIMFIT outputs a logfile containing the 11 parameters for the first image, then the values for the total flux, B/T, and sky background of the galaxy in the second filter. It also lists the χ^2_ν, $r_{1/2}$, asymmetry and symmetry indices, and concentration for the galaxy in both filters.

3.4 Decreasing the Runtimes

As these fitting routines are extremely cpu-intensive program (ask anyone who's ever used them), it takes a long time to fit large galaxy samples. For example, a 50×50 galaxy thumbnail takes about 30 minutes to run on a Pentium II 400 Mhertz processor. A galaxy sample of 200 such objects takes four days to fit with GIMFIT.

After running some tests, I found a few ways of reducing GIMFIT's run-time without affecting the results much (meaning changes are smaller than the given errors); note, this depends strongly on the data being used. All of these are applicable to the three fitting routines.

- *Decrease nsamp:* The default value is 300; change it to 200. Decrease to smaller value at your own risk.
- *psftype=delta:* Change the profile fitting so that it does not convolve the model with the PSF. This is possible only for large, bright galaxies which are well-sampled. In this case, GIM2D does not need a PSF. Remember to set *in_psf* in GIMFIT to the name of the object, i.e. gx*y*.fits.
- *earea=10:* The default value is 15; change it to 10. This decreases the size of the galaxy's thumbnails. As GIM2D fits three components (two for the galaxy, one for the sky), reducing the number of pixels decreases the run-time.

- *Undithered Images:* With WF images, use the 0.1"/pix resolution, i.e. do not oversample the PSF. Increasing the resolution may not improve the output models significantly but the run-time is proportional to the resolution.

4 Simulations

A really useful component of GIM2D is its ability to generate fake data sets and analyze them in the same manner to produce confidence limits on the models. Given a range of values for the 12 parameters, plus some for adding HII regions, GSIMUL and C2GSIMUL create galaxies from random selections of these values. The three fitting routines can then be used, with SExtractor and TinyTim, in the same manner as with the real data to fit models to the galaxies.

4.1 GSIMUL

As with GIMFIT, GSIMUL works in one filter at a time.

1. Clip a blank area from the original image that is approximately $earea \times iso\_area_{max}$. This will be the *tmpl_image*. Also have the appropriate PSF for this region.

2. Have a range file, "deV.dat" in this example, listing all of the following in **one line**:

- total flux min & max
- B/T min & max
- r_e min & max
- e min & max
- r_d min & max
- i min & max
- Sersic index n min & max
- ϕ_b
- ϕ_d
- dx
- dy
- background db
- number of HII regions (n_{HII})

- fraction of light contributed by the HII regions ($HIIfracL$) min & max
- maximum radius to place HII regions ($HIIr_{max}$).

For example,

750 284594 0 1 0.4 20.0 0 0.7 0.2 20.0 0.5 85.0 4 4 2 190 0.5 0.5 3.0 0 0 0 0

creates de Vaucouleurs bulge+exponential disk galaxies.

3. Edit GSIMUL:

```
(range_file = "deV.dat")  Name of range file
(tmpl_image = "tmpl")     Name of template image
(out_file = "gsim.out")   Name of output file
(sexroot = "gsim")        SExtractor file root name
(xgalroot = "gsim")       Extracted image root name
(reduce_cl = "gsim.cl")   Name of output GIMFIT2D cl script
(imscale = 1.)            Image scale (arcsecond/pixel)
(psftype = "tiny_wfpc")   PSF type (delta|gaussian|tiny_wfpc|tiny_nic|tinytim|user)
(in_psf = "psf814")       Name of tinytim/user PSF image
(c_abs = 0.)              Disk internal absorption coefficient (0-1)
(seeing = 1.)             Seeing FWHM (arcseconds)
(egain = 7.)              Effective gain (electrons/DU)
(earea = 10.)             Extraction area in multiple of isophotal area
(ngal = 500)              Number of galaxy simulations
(seed = -26733)           Random generator seed
(loggen = yes)            Generate flux and radius values in log (yes/no)
(osamp = 5)               Core oversampling factor
(mode = "q")
```

All the fake galaxies will have the gsim prefix and be numbered from 0. GSIMUL generates fake galaxies and their segmentation images, as well as an file listing all the parameters for the fake galaxies ("gsim.out").

Troubleshooting: If GSIMUL complains about not finding "sex.alpha" or "sexreduce.alpha," go into t_gsimul.x and change these to the correct SExtractor executable command (e.g. "sex") and whatever you used to convert the SExtractor catalogs into the .gfxt files.

4. Run GIMFIT with GSIMUL's output cl script.

4.2 C2GSIMUL

C2GSIM uses the simulated galaxies from GSIMUL to create galaxies in a second filter.

1. Create galaxies in the red filter with GSIMUL. If they do not have the filter extension in their name, rename them (e.g. gsim*n*I.fits and gsim*n*I_m.fits where *n* is the galaxy number).

2. Clip a blank area from the original image in the second filter that is approximately $earea \times iso\_area_{max}$. This will be the *tmpl_image* in C2GSIM. Also have the appropriate PSF for this region and filter.

3. Edit C2GSIM:

```
(in_list = "@gsim.lst")  List of input image(s)
(tmpl_image = "Vtmpl")   Name of template background image
(psftype = "tiny_wfpc")  PSF type (delta|gaussian|tiny_wfpc|tiny_nic|tinytim|user)
(in_psf = "psf606")      Name of tinytim/user PSF image
(egain = 7.)             Effective gain (electrons/DU)
(b1_name = "I")          b1 bandpass name
(b2_name = "R")          b2 bandpass name
(b21_zeropt = 1.26)      b21 zeropoint
(b21_blgmin = 0.5)       Bluest b2-b1 bulge color
(b21_blgmax = 1.6)       Reddest b2-b1 bulge color
(b21_dskmin = 0.5)       Bluest b2-b1 disk color
(b21_dskmax = 1.6)       Reddest b2-b1 disk color
(exptime_b1 = 3600.)     b1 exposure time in seconds
(exptime_b2 = 3600.)     b2 exposure time in seconds
(seed = -23145)          Random generator seed
(mode = "q")
```

in_list: List of galaxies without the filter suffix, e.g. gsim0, gsim1, gsim2, etc.

b1_name: Suffix for galaxies in red filter, e.g. I.

b2_name: Suffix for galaxies in blue filter, e.g. R.

b21_zeropt: Difference in the magnitude zeropoints, e.g. $R_{zpt} - I_{zpt}$.

4. Create the C2GIMFIT input files and cl script as before.

4.3 TBA

- GASYMUL
- GSELFUNC