Short manual of the *multiking_addACS* procedure Version 1.0

Maurizio Paolillo

July 26, 2007

Abstract

This document describes the use of the IDL *multiking* procedures to generate and add simulated PSF or Globular Clusters to HST ACS observations, simulating the effect of dithering on the final drizzled and stacked image. The script can be easily modified to add any type of object.

1 Introduction

This task is based on the IDL procedure *multiking_addACS.pro*, aimed at generating and add simulated PSF or Globular Clusters to HST ACS .flt images. The final images can be combined with the multidrizzle package as regular ACS images. Alternatively the shell *multiking_wrapper.csh* allows to simulate a list of objects in given positions using *multiking_addACS.pro*, add them to a set of dithered observations and combine (drizzle) the individual observations into the final image.

Required files:

- *multiking_wrapper.csh*: wrapper script to simulate a list of objects in several dithered observations and combine them in a final drizzled image. Uses *multiking_addACS.pro*.
- *multiking_addACS.pro*: IDL procedure which creates multiple GCs and adds them to a blank or real HST ACS .flt image
- *kingmod_addacs.pro*: IDL procedure which creates the individual noisy and PSF-convolved GC images to be passed to *multiking_addACS.pro*
- *makeHSTpsf.csh*: c-shell to create HST PSF with TinyTim with the appropriate spectrum and detector position

- *get_dithered_coords.csh*: c-shell to invoke the pyraf tran task to convert between distorted (flt) and undistorted (dth) coordinate grid
- *elliptical_spec.dat*: spectral template for TinyTim in ASCII format (see TinyTim help for more informations)
- final dithered image for astrometric reference
- .idc and .dxy distorsion files from the archive, *coeff?.dat files produced during the drizzling phase to be used by the pyraf tran routine
- *mask?.dat files produced during the drizzling phase if adding to original image, to allow to apply the same masking

Required software:

- *IDL*: (I tested using version 6.0 on a Linux system)
- *Pyraf* and the **Dither** package: including the *tran* routine within the Dither package to convert .flt pixel coordinates to world coordinates, and the *drizzle* task to stack the dithered observations.
- OPTIONAL: *TinyTim* HST PSF simulator (I tested using version 6.1) can be used if you don't want to use the Anderson PSF libraries. However this requires to manually uncomment part of the IDL procedure and is discouraged since it is not optimized for charge diffusion effects.

Output:

1. simulated HST ACS image, named according to the IDL input
2. simulated GC parameter list, named according to the IDL input
3. log file of the makeHSTpsf.csh and TinyTim output: makeHSTpsf.log
4. OPTIONAL: simulated PSF at each GC position are produced as files: king_*_*_*_psfconv.fits, but deleted on successful completion of the procedure. To preserve temporary PSF-convolved GC fits images comment this line in *multiking_addACS.pro*:
   `spawn, 'rm king_*_psfconv.fits' ; remove PSF-convolved GC fits images`
5. OPTIONAL: temporary TinyTim file: tmp.par (see discussion on *Tinytim* above)

2 Usage

2.1 Simulating and drizzling with *multiking_wrapper.csh*

To generate a simulated drizzled ACS image starting from a list of object positions and a set of ACS observations (including the final dithered file) run the *multiking_wrapper.csh* script followed by the input parameters and files:

Input parameters explanation:

1) File containing list of .flt files on which you want to add simulations
2) Reference drizzled image for astrometry (usually obtained with .flt files as in #1)
3) Rootname of output drizzled image with simulated objects (simulated .flt files will be named accordingly)
4) Simulated .flt should be created as orig.flt+simul [a] or just blank+simul [n]?
5) Create GC [gc] or stellar [psf] objects?
6) Generate random sample whithin assigned range [p] or read obj.params from file [r]
7) Output filename with simulated objects parameters
8) If (6)=p: numbero of objects to simulate;
   If (6)=r: File with parameters of objects to simulate Note that input parameter file requires specific format as included template anderson_coords_wcs_center.lst, i.e. identical to the output file with simulated objects parameters of #7

9-11) If (6)=p: range [min max] of core radius (pixels), conc.index and mag of objects to simulate

Examples:

1. Generate a blank drizzled image with a grid of PSFs at Anderson library coordinates.

   This example allows to create the best PSF library using Anderson templates but further accounting for your specific dithering pattern. Intermediate position PSFs can be obtained interpolating the resulting grid. Alternatively, PSF at specific positions can be produced, but the code just uses the nearest Andersen template instead of interpolating first, so the previous approach is more robust:
   > *source multiking_wrapper.csh flt.lst N1399_center_drz_sci.fits N1399_psf_center n psf r psf_simlist.lst anderson_coords_wcs_center.lst | tee multiking_wrapper.log*

2. Add 10 simulated GCs to an actual ACS dithered image:
   > source multiking_wrapper.csh flt.lst N1399_center_drz_sci.fits N1399_GC_center a gc p GC_simlist.lst 10 0.01 5.0 0.1 20 21 27 | tee multiking_wrapper.log

2.2 Simulating individual images with *multiking_addACS.pro*

To produce a single ACS image with simulated GC using *multiking_addACS.pro*:

> idl
IDL> .run multiking_addACS.pro

then provide the required parameters. Standard templates are provided in the following examples.

Examples:

1. Simulate 4 GCs and produce a blank image with simulated objects:

   IDL> .run multiking_addACS.pro
   Reference ACS flt image? j8zq07a1q_flt.fits
   Final dithered reference ACS image (_drz_sci file)? N1399_sw_drz_sci.fits
   Output simulated ACS image with GC? ACS_GC.fits
   Add simulation to original image or create new image? [a/n] n
   Perform simulations or read GC parameters from file? [p/r] p
   Output file with GC parameters for reference image? simulated_GC.lst
   Number of GC to simulate (> 1)? 4
   Core radius range in pixels [min max]? 0.05 0.2
   Concentration index range [min max]? 10 100
   Total V magniture range [min max]? 20 22

2. Read 4 GCs positions and add simulated objects to original ACS image:

   IDL> .run multiking_addACS.pro
   Reference ACS flt image? j8zq07zvq_flt.fits
   Final dithered reference ACS image (_drz_sci file)? N1399_sw_drz_sci.fits
   Output simulated ACS image with GC? ACSplusGC.fits
   Add simulation to original image or create new image? [a/n] a
   Perform simulations or read GC parameters from file? [p/r] r
   Input file with simulated GC parameters? simulated_GC.lst
   Output file with GC parameters for reference image? simulated_GC2.lst

3 Limitations

- Photometric filter is read from input image but counts/flux conversion is hardcoded in the procedure and must be changed manually in *kingmod_addacs.pro* if filter changes. The default is F606W. **MUST UPDATE**

- The scripts have been developed for specific purposes and thus the paths and several parameters are not generalized but still hardcoded into the software.

- The accuracy of the code depends on the accuracy of the Anderson PSF library (or of *TinyTim*) in simulating the actual ACS PSF, depending on which is used to generate the simulated PSF. Current version uses Anderson lib, but the code includes a section to use *TinyTim* (must be edited manually)

- PSF is created at 1:1 resolution and is centered on pixel; this can be upgraded using oversampled PSF to improve accuracy. Anyway the effect is a second-order one since this affects the convolution (unless you are producing template PSFs). **UPDATE:** Anderson PSFs templates allow for decentering, but I don't remember if this is considered when adding to dithered .flt files **(MUST CHECK)**.

- initial X,Y pos within pixel is hardcoded in the procedure to be at the center of a pixel when performing simulations, derived from Ra,Dec if reading.

- In the current version the drizzling phase of multiking_wrapper.csh uses the same weights of the reference drizzled image. Should be modified to account for different weighting if you want to accounts also for the simulated object; alternatively images can be drizzled manually.

- Due to ACS conventions det1 is upper one, det2 is lower one, but the fits extensions in the final image follow the opposite rule.