tinier-tim Documentation

Release 0

Ji Won Park

Sep 02, 2021

Contents

Contents:

CHAPTER 1

How to simulate PSFs with Tiny Tim

1.1 Step 1

The following sequence of commands will generate a PSF with the requested conditions. First, choose the name of the PSF, e.g. *psf0*

```
$tiny1 psf0
```

When prompted, choose the desired instrument, e.g. 23 for the WFC3/IR channel.

```
$ 23
```

For the position, we can choose the middle-ish pixel, i.e.

```
$550 550
```

Next is the filter passband corresponding to the instrument, e.g. F160W for the WFC3/IR channel we chose.

```
$f160w
```

We will then select the spectrum from which 13 wavelengths will be used to generate the PSF. Let's select one from a list.

```
$1
```

A table of stars and their colors will be printed. Let's choose AOV.

```
$7
```

For the PSF diameter, simply choose a reasonable diameter that will fit within your frame, e.g.

```
$5
```

I don't know what the focus, secondary mirror despace parameter does. Let's just not use it.

```
$0
```

Choose a rootname of PSF image files.

```
$test
```

The first step is complete!

1.2 Step 2

The second step is simply calling *tiny2* followed by the PSF name.

```
$tiny2 psf0
```

tiny2 will tell you that the intermediate PSF dimensions are 108 by 108, and proceed to compute the PSF at 13 wavelengths. It outputs an intermediate file called *<root name>00_psf.fits*, e.g. *test00_psf.fits* in our case, as well as a template optional parameter file required by *tiny3* called *<root name>.tt3*, e.g. *test.tt3*. The intermediate PSF is an undistorted PSF with the pixel size of 0.046241 arcsec.

1.3 Step 3

tiny3 resamples and distorts the intermediate PSF from *tiny2*. If we run:

```
$tiny3 psf0
```

with no arguments, the output PSF, named *<root_name>00.fits* (*test00.fits* in our case), is 46 by 46 pixels. But say we want to apply a super-resolution (subsampling) factor of 4. Then run:

```
$tiny3 psf0 sub=4
```

This will yield a PSF with 184 (46 times 4) pixels on each side.

Now we are ready to rebin. See *How to rebin and drizzle PSFs*.

CHAPTER 2

How to rebin and drizzle PSFs

1. Rebinning is like applying a low-pass filter, by which the pixels within the "kernel" become averaged (or summed–but the normalization doesn't matter as the PSF becomes normalized to unity in the end). Depending on the location of kernels within the bigger PSF grid, we can have various rebinned PSFs.

See Fig 2 in Ding et al 2016. Moving the blue frame around simulates dithering. This amounts to summing smaller pixels within each blue cell. If we do this 8 times, we end up with 8 rebinned PSFs.

Run

```
$python tinier-tim/rebin_psf.py <path to the PSF fits file output by tiny3> --save_
↪dir <desired working folder>
// from our Tiny Tim tutorial,
// python tinier-tim/rebin_psf.py test00.fits
```

This script creates 8 differently-binned images, named *non_drizzled_psf-[0-8].fits* in the <desired working folder>. If the *–save_dir* argument is not provided, it will create a folder named *rebinned_dir* instead.

2. Initialize IRAF within the working folder.

```
$cd <desired working folder>
$mkiraf
```

When prompted for the terminal type, enter *xterm* for a standard Ubuntu system.

This will output a config file *login.cl* accepted by IRAF. Check your device's information, and locate the section containing the list *tv*, *utilities*, *noao*, and *vo*. After *vo*, add *stsdas*, *analysis*, *dither* in separate lines.

3. Modify the drizzling config file, *dri_psf.cl*.

Note that drizzling can change the center flux by a factor of 4! If the original PSF had 61 pixels of pixel size 0.13", we expect a finer PSF with target pixel size of 0.08" to have 99 pixels. In *dri_psf.cl*, the parameters *drizzle.outnx* and *drizzle.outny* should be greater than 99.

The *drizzle.scale* parameter is a fraction of post- and pre-drizzle pixel sizes, i.e. 0.08"/0.13" ~ 0.615.

Drizzling is slapping a lower-resolution PSF on a higher-resolution grid, defined by *drizzle.pixfrac*. So *drizzle.pixfrac* should be slightly greater than *drizzle.scale.*

4. Run *drizzle.sh*.

CHAPTER 3

tinier-tim

Simulating drizzled HST PSFs.

CHAPTER 4

Installation

0. Follow the instructions on the STScI-supported documentation to create the *iraf27 conda* environment and activate it. This amounts to running the following:

```
$conda config --add channels http://ssb.stsci.edu/astroconda
$conda create -n iraf27 python=2.7 iraf-all pyraf-all stsci
$source activate iraf27
```

When you run *tiny1* on terminal, you should see

```
>> tiny1 paramfile [major=x minor=y angle=z] [jitter=x] [ebmv=x] [Av=x] [wmag=x]
```

1. Clone the editable version of this repo (not available yet!).

```
$git clone https://github.com/jiwoncpark/tinier-tim.git
$cd tinier-tim
$pip install -e . -r requirements.txt
```

2. (Optional) To run the notebooks, add the Jupyter kernel.

```
$python -m ipykernel install --user --name tinier-tim --display-name "Python (tinier-
↪tim)"
```

CHAPTER 5

Tutorials

See the documentation for tutorials.

CHAPTER 6

Feedback

Suggestions are always welcome! There is quite a bit of omission in the tutorials owing to my lack of background knowledge. If you encounter issues or areas for improvement, please message @jiwoncpark or make an issue.

CHAPTER 7

Attribution

The package heavily uses the C package Tiny Tim developed by John Krist, with additional support provided by Richard Hook and Felix Stoehr. The rebinning and drizzling code was written by Xuheng Ding (@dartoon) and modularized with minor modifications by Ji Won Park (@jiwoncpark). The tutorials were also written by Ji Won Park (@jiwoncpark).