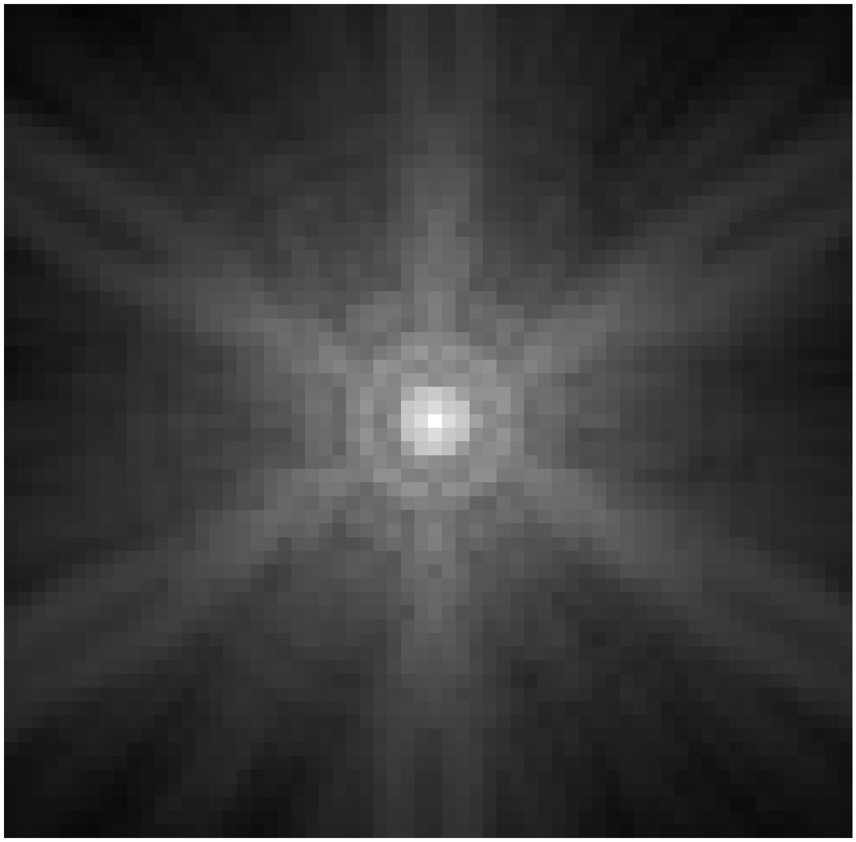
Tiny Tim / Spitzer User's Guide



Version 2.0
June 2006

Developed by John Krist
For the Spitzer Science Center

Get help from : ***help@spitzer.caltech.edu***

Contents

Changes in Version 2.0

This is the first release since the launch of Spitzer, and it incorporates updates based on measurements of on-orbit data, including:

- The orientations for many of the instruments have changed and are intended to match those of the BCD files.
- The plate scales have been updated to the nominal BCD data set scales at the field centers of the detectors.
- Non-square pixels have been implemented (used for MIPS).
- The telescope secondary mirror support vane parameters have been updated, though the resulting diffraction spikes are still somewhat different than what is seen on-orbit.
- Aberrations for the IRAC, MIPS 24, and IRS peakup cameras have been measured by iteratively fitting Tiny Tim models to stellar images while altering parameter values.
- The charge diffusion kernels for IRAC have been removed, as the on-orbit data do not appear to have significant diffusion when compared to the no-diffusion models.

Caveats

Geometric distortion is not included. This should not affect PSF cores significantly, but may come into play when PSFs are computed covering large areas. The orientations and scales are intended to match those at the centers of the BCD files.

The modeled diffraction spikes tend to depart from reality past a dozen or so Airy rings, due to the complicated structure of the secondary mirror support vanes.

Detector intrapixel sensitivity is not modeled.

While the aberrations at the center of the field have been updated to match the on-orbit data for some cameras, the field-dependent aberrations have not been measured. The pre-launch predicted variations will continue to be used.

Tiny Tim creates models intended to match the pixel scale and orientation of stellar PSFs as they appear in BCD files (excluding geometric distortion). The scales and orientations of post-processed data (mosaics) are different. In order to match to such post-BCD images, the Tiny Tim models will need to be processed in the same manner.

Acknowledgements

The author thanks Karl Stapelfeldt, Lisa Storrie-Lombardi, Susan Stolovy, David Ardila, Patrick Lowrance, John Stansberry, Debra Padgett, and Luisa Rebull for their assistance.

Introduction

Tiny Tim is a program for computing point spread function (PSF) models. It was originally designed for simulating Hubble Space Telescope (HST) PSFs and has been in wide use by the HST community since it was introduced in 1991. At the request of (and under contract to) the Spitzer Science Center, a new version was developed in early 2000 specifically for Spitzer. Only Tiny Tim/Spitzer is described here; the HST version, which is a completely separate program, can be obtained from http://www.stsci.edu/software.

Tiny Tim/Spitzer is written in standard C and is distributed as source code. It has been successfully compiled on a variety of systems, including UNIX, Microsoft Windows, and Linux. It's not fancy – there are no widget interfaces or fancy graphics. It simply asks a few basic questions and then goes off and computes the PSF model.

Tiny Tim/Spitzer *is not* an optical ray-tracing program, like Code V or Zemax. Those programs are used to design optical systems and determine their performance. Tiny Tim is hard-coded to easily model PSFs for specific optical systems, using parameters (aberrations, vignetting properties, etc.) derived from ray tracing software and measurements of on-orbit data. Information from other sources, like spectral sensitivities and detector characteristics, may also be included.

Questions and comments should be directed to **help@spitzer.caltech.edu.**

Uses of Model PSFs

Model PSFs have been used for a wide variety of applications:

- **Proposal planning**: predict limiting sensitivities and resolution
- **Photometry:** PSF profile fitting by the shifting and scaling of models; testing completeness by generating synthetic images
- **Algorithm testing:** photometric, deconvolution, and dithering algorithm testing
- **PSF subtraction**
- **Data modeling:** convolution of model data with model PSFs to simulate observed images

PSFs from the HST version of Tiny Tim have been used for all of the above. It has been especially useful for algorithm testing, since the PSFs are noiseless and can be generated at subsampled resolutions.

Spitzer

A diagram of the Spitzer focal plane layout is given in Figure 1, with the assumed detector orientations shown.

As with any complex optical system, the aberrations in the Spitzer instruments will vary with field position. Off-axis astigmatism and field curvature are traits of the Ritchey-Chretien optical system used for the Spitzer telescope, though coma is (ideally) corrected over the field. The instruments themselves also introduce their own field-dependent aberrations, including defocus, astigmatism, and coma. These variations have been characterized using the Code V ray-tracing package based on the instrument design parameters and are included in the Tiny Tim models for the IRAC and MIPS imaging modes. They are not included in the spectrograph models due to the limited fields of those instruments (though on-axis aberrations for each spectrograph are included). In general, the changes in the PSF over the field of view caused by these aberrations are probably unimportant in most cases.

In addition to classical, low-order aberrations such as coma and astigmatism, there are higher-frequency wavefront errors caused by imperfections in the telescope mirrors resulting from the polishing process. These zonal aberrations are included in the Tiny Tim/Spitzer models for the primary and secondary mirrors. The effect of these errors is to diffract light further out into the PSF wings, increasing the local background. Because wavefront errors are more prominent at shorter wavelengths, the IRAC cameras, especially at 3.5 μm, are the most affected.

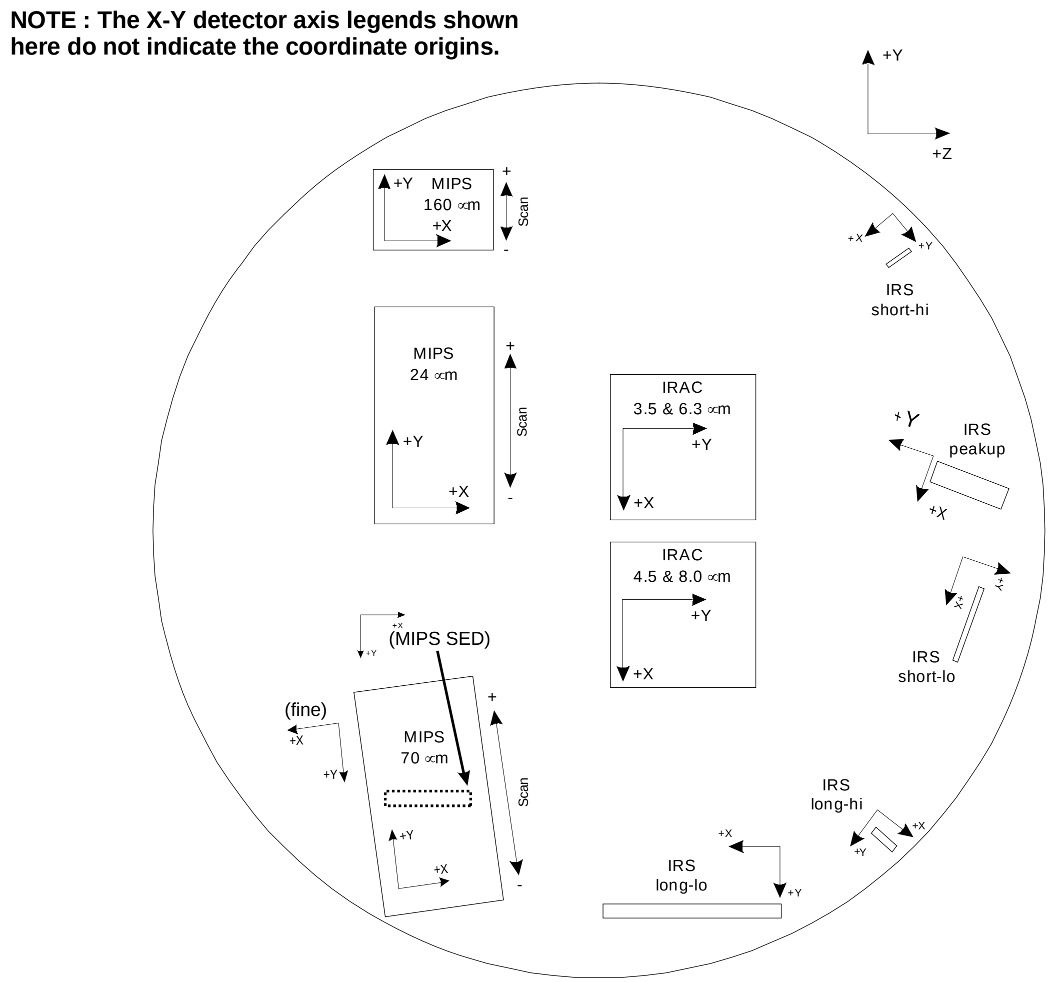


Figure 1. Spitzer Focal plane layout (projected onto the sky). The X-Y detector axes assumed by Tiny Tim are shown. The sizes and placements of the arrays as shown are not exact.

The Spitzer obscuration pattern also varies with field position. In all instruments, this pattern is defined by the primary mirror and the blades that support the secondary mirror assembly. The central obscuration in the system is not the secondary mirror but actually the central mask on the primary. The beam of light from the primary mirror will strike the secondary at different angles depending on the field position of the observed object. Since the secondary support blades are between the primary and secondary mirrors, the beam will also strike those blades at different angles, effectively changing their position and shape in the obscuration pattern (see Figure 2). The net effect is that the PSF diffraction spikes will change with field position. These changes have been characterized using ray-tracing software developed by the author, based on the as-designed dimensions of the blades. The change in the PSF due to obscuration field-dependence is typically greater than that caused by aberration variations.

It should be noted that Tiny Tim/Spitzer projects all obscurations into a single pupil plane and assumes that far-field diffraction occurs only in that plane. In the case of the Spitzer secondary support, however, diffraction occurs continually along the blade edges, from the primary to the secondary. The algorithms used by Tiny Tim cannot simulate this effect, so their effects must be approximated by single-plane diffraction. Comparisons with on-orbit data indicate that while Tiny Tim recreates the diffraction spike patterns reasonably well in general, the spikes do not fully line up with the observed ones, and there are small structures that simply do not agree well. This is a limitation of the modeling process.

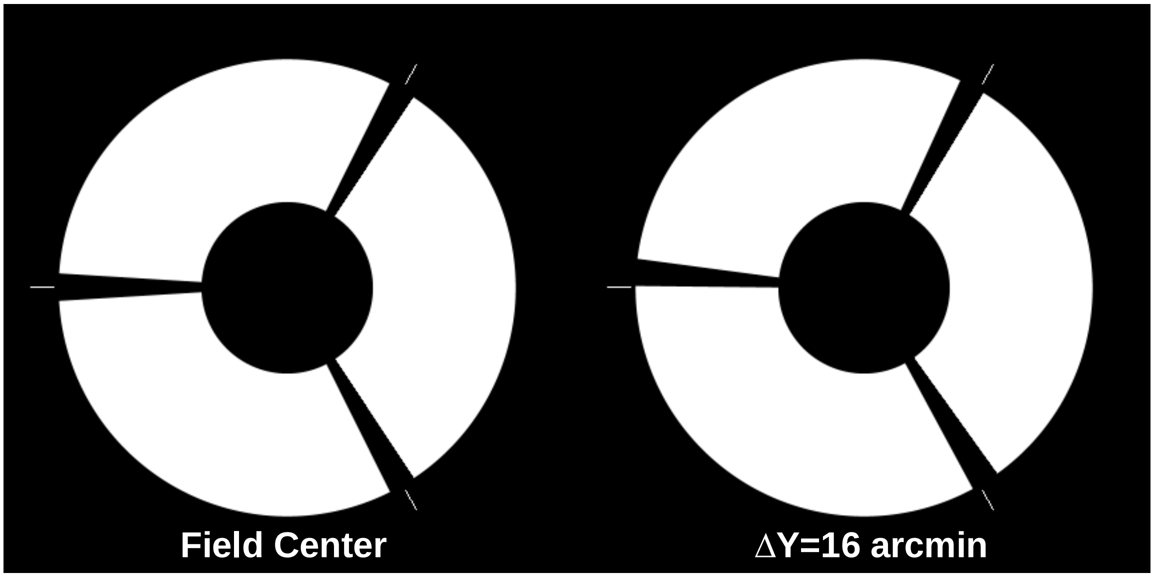


Figure 2. Spitzer obscuration patterns for on-axis and 16 arcmin off-axis field angles. The tapered shape of the spiders is due to the secondary support blades being in the converging beam from the primary.

General Caveats

Model PSFs do have significant limitations, though some of these are shared with observed PSFs as well. The simulation is only as good as the input parameters, which are often inaccurate due to our limited knowledge of the as-built (rather than as-designed) optical system. A large number of factors affect the PSF, including:

- Optical and detector responses with wavelength
- Object spectrum
- Focus (possibly time and position dependent)
- Aberrations (which may depend on alignment, focus, and position)
- Polishing errors in the mirrors
- Obscuration positions and sizes (field dependent in Spitzer)
- Detector orientation (rotation, pixel size)
- Detector subpixel response
- Geometric distortion
- Scattering by dust, scratches, chips, rough surfaces

Many of these may vary with time and/or field position, so it is impossible to create a model that will match an observed PSF pixel for pixel. Of course, even observed PSFs will often have significant differences due to these same variations.

Tiny Tim creates models intended to match the pixel scale and orientation of stellar PSFs as they appear in BCD files (excluding geometric distortion). The scales and orientations of post-processed data (mosaics) are different. In order to match to such post-BCD images, the Tiny Tim models will need to be processed in the same manner.

Installation

Compiling Tiny Tim

Tiny Tim is provided as source code and ASCII text tables in a gzip-compressed tar file. This file should be downloaded in binary mode and then unpacked into a separate directory on your machine, like so:

```
gunzip stinytimv2.0.tar.gz
tar xvf stinytimv2.0.tar
```

After this, the program needs to be compiled. A `Makefile` is provided to automate this task, with default compiler settings for Solaris and Linux versions of UNIX. For other systems, you should edit the `Makefile` and change the compiler settings for the OTHER mode as necessary. The programs are compiled by running make with one of the following commands:

```
make sparcstation
make linux
make hp
make other
```

The appropriate pointers to the Tiny Tim programs and directory must be included in your environment file (.cshrc, .profile, or whatever is appropriate to the system/shell).

The following shows a c-shell example :

```
setenv STINYTIM /usr/local/stinytim
alias stiny1 $STINYTIM/stiny1
alias stiny2 $STINYTIM/stiny2
```

If you are running the Bash shell (the default for most Linux installations), you can follow this example for entries in your .bash_profile file (set the STINYTIM directory to whatever is appropriate for you) :

```
STINYTIM=/usr/local/stinytim
export STINYTIM
alias stiny1='$STINYTIM/stiny1'
alias stiny2='$STINYTIM/stiny2'
```

Multithreading

Tiny Tim supports multithreaded execution on multiprocessor systems that support POSIX threads (Pthreads). This includes Solaris 2.5 and later, and current distributions of Linux. Only a specific portion of Tiny Tim has been modified to support multithreading - specifically, the routine that sinc interpolates the Nyquist-sampled PSF to the pixel-integrated PSF. Thus, multithreading is most effective on large PSFs, like those often generated when subsampling, in which case this routine is typically the most time-intensive portion of the code.

To compile Tiny Tim for multiprocessor systems, use one of the following `make` commands :

```
make threadedlinux
```

or

```
make threadedsolaris
```

Note to Linux users : *If you have a multiprocessor Linux system, the kernel must be compiled to support SMP (symmetric multiprocessing) to use more than one processor. If you ordered a multiprocessor system with Linux installed, it is probably already SMP enabled.*

Running Tiny Tim

Tiny Tim consists of two programs – ***stiny1*** and ***stiny2***. ***stiny1*** asks a series of basic questions and generates a parameter file that is read by ***stiny2***. ***stiny2*** does the actual PSF modeling, requiring no additional input from the user. ***stiny2*** can be run in batch mode if desired. Both ***stiny1*** and ***stiny2*** require a single command line argument – the name of the parameter file. For example:

```
stiny1 test.par
stiny2 test.par
```

stiny2 produces a FITS image file.

Command Line Parameters

stiny1 also allows the following optional command line parameters (given after the filename) :

`jitter=`*x*
: Convolve the PSF with a Gaussian kernel approximating *x* arcseconds RMS of jitter.

`wmag=`*x*
: Increase or decrease by a factor of *x* the default number of wavelengths used in computing a polychromatic PSF (i.e. `wmag=0.5` would use only half the default number of wavelengths, while `wmag=2` would double the number, more finely sampling the response curve). A minimum of one wavelength and a maximum of 200 are allowed. This has no effect when generating a monochromatic PSF.

`pfile=`*filename*
: Instrument parameter file to use instead of the default one provided in the STINYTIM directory. The default files are listed at the end of each instrument's section in this manual.

`scan=`*x*
: (MIPS only) Amount of field offset caused by a tilt of the scan mirror. See the MIPS section for more details.

For example, to generate a PSF with 0.3 arcsec RMS of jitter, you could enter :

```
stiny1 test.par jitter=0.3
```

General Input Parameters

A number of the questions which ***stiny1*** asks are common to all of the instruments and are discussed here. Inputs specific to each instrument are listed in later sections.

PSF Size

stiny1 asks you for the diameter of the PSF you wish to generate in arcseconds. From this it determines the proper grid sizes to use. The maximum possible size of a model PSF is limited by the array dimensions used to compute the Nyquist-sampled PSF. This size depends on the shortest wavelength in the bandpass.

Bandpasses/Wavelengths

You have the option of generating a purely monochromatic PSF or one that includes wavelength variations over an instrument's bandpass (for broadband instruments).

Tiny Tim has a database of wavelengths and respective sensitivities for each instrument's bandpass (including filter and detector throughputs). A polychromatic PSF is created by generating monochromatic models at each wavelength and adding them together with the corresponding weights. The wavelengths have been chosen to properly sample the bandpass. The throughput curve is multiplied by a spectrum to simulate the PSF variations caused by object color.

You also have the option of providing a list of wavelengths and weights instead of using the default curves. You can also provide your own spectrum. See Appendix B : *Object Spectra and Throughput Curves* for more information.

Sampling

Model PSFs can be generated at an arbitrary sampling. Tiny Tim will ask if you would like to generate a subsampled PSF. If you choose to do so, it will ask you for the subsampling factor, which is an integer from 1 to 10 (i.e. 5 means that the resulting pixels are 1/5 of normal size in both dimensions). Subsampling increases the computation time by roughly the square of the subsampling factor for a given PSF diameter. If you are using high subsampling factors (>5), then you may be better off subsampling by 5 and then interpolating the result.

Output Files

The model PSF will be written to a FITS image file. If you generated just one PSF, the file name will have the rootname you specified in ***stiny1*** with ".fits" appended. If you generated PSFs for more than one position (an option available for certain cameras), the filenames will end in successive numbers, beginning with "01" (e.g. `psf01.fits`, `psf02.fits`, …).

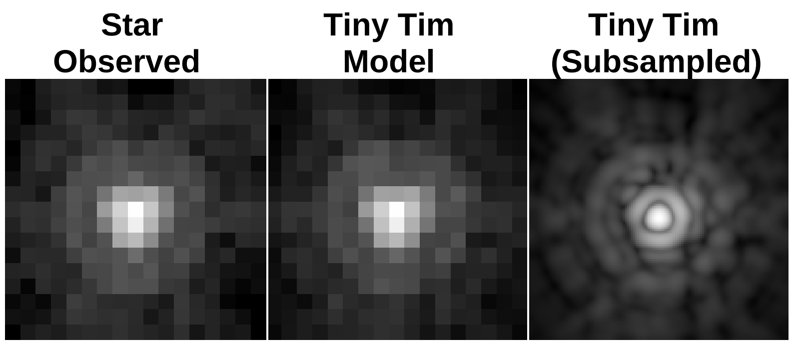
The FITS file header will contain basic information about the simulation, including the position of the PSF center on the detector, the aberrations used for that position, and the PSF pixel size.

IRAC

Please check Figure 1 for the detector-to-focal-plane mapping assumed by Tiny Tim.

IRAC (Infra-Red Array Camera) consists of four 256 by 256 pixel detectors that are dedicated to broadband imaging. The 3.5 (irac1) and 6.3 (irac3) μm arrays observe the same portion of the sky by means of a beamsplitter. The 4.5 (irac2) and 8.0 (irac4) μm arrays share a nearly contiguous region in the same way. The nominal pixel size assumed by Tiny Tim is 30 μm (1.21 arcsec/pixel at f/6). IRAC is undersampled by about a factor of three at the shortest wavelengths.

Camera	Central Wavelength λ_c (μm)	Detector Pixel Size	Nyquist Sampling Requirement @ λ_c
IRAC 1	3.5	1.21”	0.42”
IRAC 2	4.5	1.21”	0.55”
IRAC 3	6.3	1.21”	0.76”
IRAC 4	8.0	1.21”	0.97”



(Left) Star observed in IRAC channel 2 (4.5 μm); (Middle) Tiny Tim PSF model (registered to observed star position); (Right) Tiny Tim subsampled PSF.

Queries specific to IRAC

Position : The user is asked for the X and Y position of the object on the detector (integer values from 0 to 255 pixels), from which field-dependent aberrations and the obscuration pattern are determined. PSFs at multiple positions can be generated in a single run by providing the name of a file containing a list of up to 100 X-Y pairs (the filename must be proceeded by a “@” when ***stiny1*** asks for the position). The assumed mappings of the detector X-Y axes to the sky are shown in Figure 1.

Bandpass : The user is given the choice of (a) using the default bandpass curve (found in the STINYTIM directory as **`irac1.band`** – **`irac4.band`**) to generate a polychromatic PSF; (b) entering a wavelength (in microns) to generate a purely monochromatic PSF; or (c) entering the name of a file containing a user-generated bandpass curve to generate a polychromatic PSF (for the file format, see Appendix B).

Charge Diffusion in IRAC 1 & 2

Earlier versions (before 2.0) of Tiny Tim/Spitzer would convolve the IRAC channels 1 and 2 PSFs with a 3 x 3 pixel kernel to represent blurring by (presumably) charge diffusion. Comparisons with on-orbit data show that there is no significant pixel-to-pixel blurring of this sort, so such convolution no longer is performed.

Caveats

Because IRAC observes at the shortest wavelengths of the Spitzer instruments, it is the most sensitive to aberrations, especially those caused by the polishing errors in the telescope mirrors (which are included by Tiny Tim). Measurements of aberrations using on-orbit data are included for each of the IRAC channels. These represent the aberrations at the center of the detector. Field-dependent changes in the aberrations are estimated from pre-launch ray-trace models.

Intrapixel variations are not included.

Geometric distortion over the IRAC field is not included in the models.

The instrument parameter files for IRAC are located in the STINYTIM directory as : irac1.pup, irac2.pup, irac3.pup, and irac4.pup.

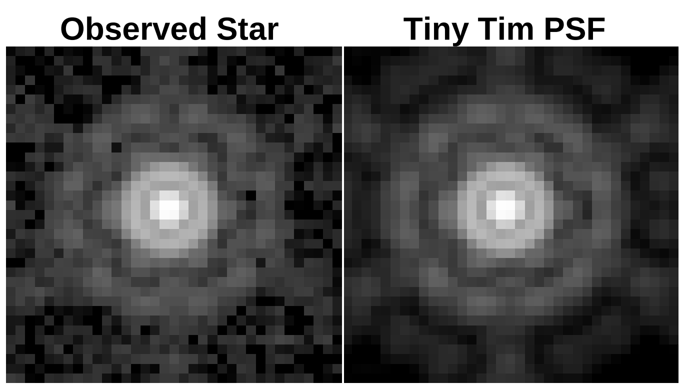
MIPS

Please check Figure 1 for the detector-to-focal-plane mapping assumed by Tiny Tim.

MIPS (Multiband Imaging Photometer for Spitzer) is comprised of three separate channels dedicated to imaging at 24 μm, 70 μm, and 160 μm. MIPS includes scanning mirrors that allow motion compensation while the telescope slews, providing imaging of larger areas of the sky than is possible with a single exposure. Subpixel steppings of the scan mirrors and telescope can also provide for enhanced resolution using dithering/drizzling algorithms. The 24 μm channel has a 128 × 128 pixel detector that covers a 5.3' × 5.3' field (5.3' × 8.1' in scan mode). The 70 μm camera uses a 32 × 32 pixel detector and has two selectable resolutions. In wide-field (survey) mode, it covers 5.3' × 5.3' (5.3' × 8.1' when scanning); in narrow-field (superresolution) mode, it covers 2.6' × 2.6' (2.6' × 4.1' scan). The 70 μm channel also includes a low-resolution spectrographic (SED) mode with a 20" wide slit. The 160 μm channel uses a 20 × 2 pixel array that covers 5.3' × 3.4' in scan mode.

Camera	Central λ_c (μm)	Detector Pixel Size[1]	Nyquist Sampling Requirement @ λ_c
MIPS 24 μm	24	2.49" x 2.60"	2.9"
MIPS 70 μm WF	70	9.86" x 10.07"	8.5"
MIPS 70 μm NF	70	5.10" x 5.25"	8.5"
MIPS 70 μm SED	70	6.6" x 6.6"	8.5"
MIPS 160 μm	160	15.965" x 17.963"	19.4"

[1]SED mode pixel size is 1/3 slit width



(Left) Star observed in the MIPS 24 μm channel; (Right) Tiny Tim model, registered to the observed PSF position.

Command-line parameters to *stiny1* specific to MIPS

SCAN (*Used for 24 μm and 70 μm imaging modes only*) **:** Specifies the offset position, in arcseconds, of the camera field due to a tilt of the scan mirror. If not given, then the camera field is assumed to be centered with a scan offset of zero. The scan direction is along the axis angle given in the table, with positive offsets towards the +Y CTA direction (see Figure 1). The scan offset is used to compute scan-dependent variations in the aberration parameters, as well as changes in the obscuration pattern due to field position. The field position of a PSF is computed from the scan position and the specified detector coordinates.

Queries specific to MIPS

Position *(24 μm and 70 μm imaging)* **:** The user is asked for the X and Y position of the object ***in detector pixels*** (integer values from 0 to *n*, where *n*=127 for 24 μm and *n*=31 for 70 μm). These are used to compute the field-dependent aberrations and obscuration pattern. PSFs at multiple positions can be generated in a single run by providing the name of a file containing a list of up to 100 X-Y pairs (the filename must be proceeded by a "@" when ***stiny1*** asks for the position). The assumed mappings of the detector X-Y axes to the sky are shown in Figure 1, and the detector rotations are given in the table above.

Position (*160 μm*) **:** The user is asked for the X and Y field position offsets (floating-point values) of the object ***in arcseconds from the detector center at the default scan position***. These values are used to compute the field-dependent aberrations and obscuration pattern. Because the 160 μm detector is, for practical purposes, a one-dimensional array, the X coordinate is translated to a detector pixel position while the Y coordinate is effectively a scan position. See the description above on how to generate PSFs at multiple positions.

Bandpass : The user is given the choice of (a) using the default bandpass curve (found in the STINYTIM directory as **`mips24.band, mips70wf.band,`** etc.) to generate a polychromatic PSF; (b) entering a wavelength (in microns) to generate a purely monochromatic PSF; or (c) entering the name of a file containing a user-generated bandpass curve to generate a polychromatic PSF (for the file format, see Appendix B).

Caveats

The scan position affects the PSF in two ways. First, the aberrations vary with the scan mirror angle. And second, an change in the camera's field position due to a scan offset results in a different obscuration pattern. However, both of these effects are small, and minor uncertainties in the user-provided scan position or in the scan axis angle assumed by Tiny Tim are insignificant.

The field position entered by the user is used to compute the field-dependent aberrations and obscuration pattern. These vary slowly over the MIPS field, so the user should not be worried about providing a position accurate to a pixel or anything like that. The scan- and field-dependent aberration relations were derived from ray tracing, and thus reflect the as-designed values. Deviations in the real optical system will result in somewhat different aberrations. Because MIPS includes a number of tilted components, the aberration relations are likely to be sensitive to any misalignments.

The 70 μm SED mode slit width is 1/2 the diameter of the first dark PSF ring, so the throughput is essentially insensitive to field-dependent aberrations. Therefore, the user is not asked for a field position for this mode. The default sampling of the PSF was chosen so that 3 pixels span the width of the slit (19.8"). The long axis of the slit would be oriented along the horizontal axis of the model.

The PSFs produced by Tiny Tim do not include any geometric distortion, and the pixels are assumed to be the same size at all field positions and scan settings.

Blue leak in the 160 μm channel is not included.

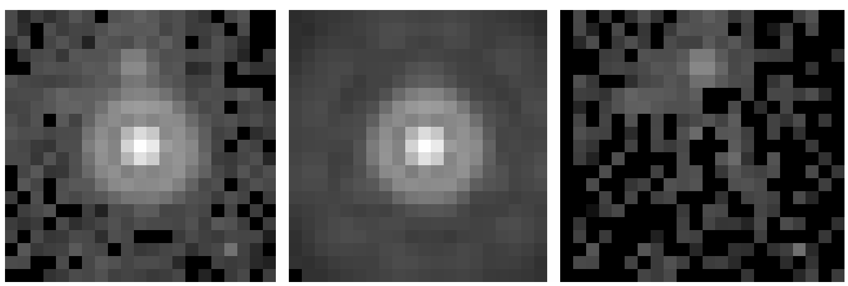
The instrument parameter files for MIPS are included in the STINYTIM directory as : mips24.pup, mips70wf.pup, mips70super.pup, mips70sed.pup, and mips160.pup.

IRS

The Spitzer InfraRed Spectrograph (IRS) provides four low and medium-resolution spectrographic modes from 4 to 40 μm. It also has two small imaging arrays centered at 15 μm and 23.5 μm that provide acquisition capabilities.

Spectrographic Mode	Wavelength Range	Slit Size	Tiny Tim PSF Default Pixel Size	Nyquist Scale (λ_{min}-λ_{max})
Short-Low	4 – 15 μm	3.6” × 57”	1.20”	0.5” – 1.8”
Short-High	10 – 19 μm	4.7” × 11.3”	1.57”	1.2” – 2.3”
Long-Low	15 – 40 μm	10.5” × 168”	3.50”	1.8” – 4.9”
Long-High	19 – 37 μm	11.1” × 22.3”	3.70”	2.3” – 4.5”

Peakup Array	Central Wavelength	Pixel Size	Nyquist Sampling Requirement @ λ_c
Peakup (Blue)	15 μm	1.835”	1.82”
Peakup (Red)	23.5 μm	1.835”	2.85”



(Left) Observed star on IRS blue peakup array; (Middle) Tiny Tim PSF, registered to align with the observed star; (Right) Model PSF subtracted from observed star.

Tiny Tim will ask for the wavelength of the PSF when the user selects one of the four spectrographic modes. For either of the peakup arrays, the user is asked for the object spectrum, and the detector's bandpass is used (though the user has the option of providing a bandpass file instead).

Tiny Tim generates two-dimensional PSFs for the spectrographs with pixels 1/3 the width of the slits. The models are oriented so that the long axis of a slit is along the horizontal direction. The image orientations are shown in Figure 1.

Measurements of aberrations in on-orbit images of stars in the blue peakup array are included. Aberrations in the red channel are assumed to be the same as those in the blue.

The instrument parameter files for IRS are located in the STINYTIM directory as : irslonglow.pup, irslonghigh.pup, irsshortlow.pup, irsshorthigh.pup, irspeakupblue.pup, irspeakupred.pup.

Appendix A

How PSFs are Simulated

The following is a brief description of how Tiny Tim simulates PSFs. For a more thorough discussion of optics and diffraction optics, check out *Astronomical Optics* by Daniel Schroeder.

A PSF is the diffraction pattern caused by light from a point source passing through an optical system, with the various optical components and obscurations adding structure to it. In the case of Spitzer, the obscurations include the inner and outer edges of the primary mirror and the secondary mirror support vanes. Optical aberrations manifest themselves as errors in the optical path length. For Spitzer these may include spherical aberration, defocus, coma, astigmatism, clover, and mirror zonal aberrations, along with others.

These properties can be simulated on a computer by creating a two dimensional array which contains the aperture function (basically a mask which shows where the beam is not being obscured) and another array containing the optical path differences (OPD) function. The OPD can be calculated as the sum of Zernike polynomials, which describe specific aberrations, and additional terms, such as local errors in the mirrors surfaces.

These two functions are generated for each point on a grid of N/2 by N/2, where N is the dimension along one edge of the arrays. This provides for Nyquist sampling of the functions. The array size depends on how large a PSF is needed and the wavelength.

The OPD and aperture function, A, form the complex pupil function:

$$P = Ae^{(2\pi i OPD/\lambda)}$$

and the Fourier transform of the pupil function is the amplitude spread function (ASF). Finally, the modulus squared of the ASF is the PSF:

$$PSF = \left| FFT(P) \right|^2$$

This is the Nyquist-sampled PSF for a specific wavelength (in which the OPD is defined), with the spacing between pixels being $F\lambda/2$, where F is the focal ratio of the instrument (such as 6.0 for IRAC). Thus, the sizes of the arrays used to generate the Nyquist-sampled PSF are dependent on the required size of the integrated PSF and the wavelength. Shorter wavelengths require larger grid sizes, and thus more time to compute.

This PSF is then Fourier transformed to produce the optical transfer function (OTF). If jitter is specified, the OTF is multiplied by a function that models Gaussian jitter.

Integration onto detector pixels of a given size is performed by multiplying the OTF by the analytical expression for the integrated detector OTF. The result is then Fourier transformed back and the integrated values sampled using sinc interpolation.

Because the PSF changes with wavelength, a number of PSFs have to be generated by the above method at wavelengths which sample the filter being used, to account for the system's bandpass. These monochromatic models are combined via a weighted addition. The weights are dependent on the optical system throughput, including the filter, and the object spectrum.

Appendix B

Object Spectra and Throughput Curves

Because the size and shape of the PSF is dictated by diffraction rather than seeing effects, you must pay attention to how the PSF varies with wavelength. As wavelength increases, the diffraction structure (rings, spikes) expands proportionally. In the same broadband filter, the PSF of a red star may be noticeably larger than that of a more blue one. If you subtract a red object PSF from a blue object's, you can end up with significant residuals (this only applies to wide band filters – the change over the bandpass of a narrow filter is far too small to be significant).

Since object color can make a difference, you should try to match the object spectrum when you generate Tiny Tim PSFs in broadband systems. You can specify a blackbody or power-law spectrum, if that is adequate, or you can provide your own spectrum.

Specifying Your Own Spectrum

If your object is so different that it cannot be reasonably well represented by one of the default spectrum equations, you also have the option of providing Tiny Tim a spectrum to use instead.

You provide an ASCII text file, which contains the wavelength (in microns) and corresponding flux, one pair per line, up to 1000 pairs. The wavelength range of your spectrum should cover the range of the bandpass you will be using and then some (otherwise, the spectrum will be extrapolated). The default bandpass files used by Tiny Tim are identified in each instrument's section in this manual.

The spectrum file must be in a specific format. The first line, beginning with the first character, must be one of the following, which describe the units of the fluxes given : FLAM, FNU, PHOTLAM, JY. These correspond to :

FLAM	ergs μm^{-1}
FNU	ergs Hz^{-1}
PHOTLAM	photons μm^{-1}
JY	Janskys (W Hz^{-1})

Since Tiny Tim is only interested in the relative fluxes between wavelengths, the normalization of the spectrum is not important. So, if you have a spectrum in ergs s^{-1} cm^{-2} $Å^{-1}$, then you can just use those values with the FLAM specifier – the units are directly proportional.

An example of the file format is :

```
PHOTLAM
1.00  0.005
1.05  0.009
1.10  0.013
```

Your spectrum *should not* include system throughput. Tiny Tim will interpolate your spectrum to determine the flux for each wavelength in the throughput curve. It then multiplies this by the throughput curve wavelength spacing to get the appropriate PSF weighting at that wavelength. Because of the interpolation, it is best if your spectrum is not highly oversampled relative to the throughput curve.

Specifying Your Own Throughput Curves

If for some reason Tiny Tim's default bandpass throughput tables are not sufficient, you can provide your own. To do so, you must create a text file containing the wavelength (in microns) and the corresponding transmission, one pair per line, in order of ascending wavelength. A maximum of 200 wavelengths can be entered. The wavelengths should be spaced by about 0.018λ.

Note that you can increase or decrease the number of wavelengths in the default throughput curves by using the WMAG keyword parameter on the ***stiny1*** command line (described earlier in this manual).

Appendix C

PRFs and Using Subsampled PSFs

Subsampling can be useful for creating PSFs at different centerings within a pixel, or for including subpixel detector effects.

A Tiny Tim PSF is always centered within a pixel (though the centroid may be offset due to asymmetries introduced by coma). If you require a PSF that isn't pixel-centered, then you need to generate a subsampled PSF, shift it, and rebin it to detector sampling. For instance, if you generated a PSF with 3x subsampling, then you can shift and rebin it to replicate a PSF centered at 1/6, 1/2, or 5/6 of a pixel in each direction.

If you need a PSF centered at some more arbitrary position within the pixel (as in the case of fitting a model to an observed PSF), you will need to subsample more and probably interpolate for even finer shifting. The amount of subsampling required is determined by the Nyquist criterion, which is listed in each instrument's section in this manual. Typically, you want to be somewhat better sampled than the Nyquist criterion to provide for reasonably accurate interpolation (use a higher-order interpolator; bilinear doesn't work well on this stuff). In the end you will still need to rebin the result to normal sampling.

Subsampled PSFs Versus Subsampled PRFs

There is a difference between subsampled PSFs produced by Tiny Tim and how they are used versus the subsampled point response functions (PRFs) derived from on-orbit data and provided for IRAC and MIPS on the Spitzer Science Center web site.

For example, each value in a Tiny Tim 5x subsampled model represents the flux that would be collected in just a 1/5 by 1/5 fraction of a detector's pixel. To determine the flux that would be measured in full detector pixels, one would need to bin 5 by 5 patches of these subsampled values.

In comparison, the subsampled PRFs on the SSC web site represent more a lookup table rather than a subsampled representation of the detector image. Each value in a 5x subsampled PRF represents the flux in a full detector pixel, with various steppings of the PSF across the detector. To get a normally sampled image, one would sample the PRF every 5 points.

Appendix D

Miscellaneous Notes

Tweaking the Parameter File

The parameter file produced by stiny1 has some "secret" flags that can produce additional output files. Since this is a plain ASCII file, you can use any text editor to alter these flag values. Be sure not to change any other values, as some parameters are tied to others or have been precomputed by stiny1. The flags are :

Write pupil map?	Setting this to 1 will cause the computed obscuration pattern to be written to **tinytim_pupil.fits.** If using multiple positions in one run, only the pattern for the first position will be written.
Write OPD map?	Setting this to 1 will cause the computed OPD (optical path differences) function to be written to **tinytim_opd.fits**. This 2D image contains the aberration map in microns of wavefront error.
Write Nyquist PSF?	Setting this to 1 will cause the Nyquist-sampled PSF for the first wavelength to be written to **tinytim_nyquist_psf.fits**. The sampling of this PSF is $F\lambda/2$, where F is the focal number.
Use primary map?	Setting this to 0 will exclude the map of the primary mirror zonal errors from the aberration function.
Adjust for field dependent aberrations?	Setting this to 0 will cause Tiny Tim to use the aberrations for the center of the field of the instrument for all positions.

Aberrations

The wavefront aberration function generated by Tiny Tim consists of the sum of Zernike polynomials and the primary mirror zonal error map. The Zernike polynomials used are normalized for a 33% central obscuration, with the RMS wavefront error in microns for each aberration specified by a coefficient to each polynomial. Note that this is different, both in order and form, from the Zernike polynomials assumed by Code V or Zemax, which give peak-to-valley errors for unobscured systems. The angle of the polynomials is defined relative to the telescope's coordinate system, going counter-clockwise from the +Z axis to the +Y axis.

The aberrations listed in the parameter file produced by ***stiny1*** are for the center of the instrument's field. ***stiny2*** computes the field-dependent aberration offset for a given position and adds this to the field-center value to produce the final Zernike coefficient. The values for the most important aberrations are written to the header of the PSF FITS file.

A defocus of the telescope (by moving the secondary mirror) can be duplicated by manually changing the focus aberration value (Z4) in the parameter file. For each micron of motion of the Spitzer secondary, the Z4 aberration in the file should change by 0.0216. The focus becomes more negative as the secondary is moved towards the primary.