New Point-Source SNR Option for Mopex/Apex: "Photerr"

Summary

There is a new option for calculating point-source-fitted flux signal-to-noise ratio, SNR, in the Apex and Apex_1frame modules of the SSC's Mopex package. It's called here the Photerr for short. It's similar to an aperture photometry error, but over a fixed box based on the size of the PSF, and with the option of switching between data uncertainties, and background noise + total source photon noise. It's an easily-calculated quantity intended to approximate the scatter of repeated flux measurements.

1. Background: The Old SNR

There are two flux uncertainties associated with point-spread-function (PSF) fitting available in the Apex extract tables, delta_flux and signal-to-noise ratio (SNR). Delta_flux is the formal uncertainty in the least-square fitted flux (e.g. Bevington 1969). Often this will underestimate the true flux uncertainty for coarsely sampled PSF's because of correlated errors (see Fig. 1 for example). The flux uncertainty is correlated with the position uncertainty and the individual parameter uncertainties don't reflect this.

An attempt was made earlier to provide a working measure of the flux uncertainty in the tables, SNR. (This also allows users to select sources by SNR.) The signal is the PSF-fitted flux. How was the noise determined? It was:

$$\sigma = \sigma_{centerpixel} \times NP \tag{1}$$

where

$\sigma_{centerpixel}$ = uncertainty in nearest pixel to peak

NP = ratio of the total centered PRF volume to that under the central pixel

For uncertainties on the right in data units, e.g. MJy/sr for Spitzer, the final σ was converted to flux units.

This approximates the flux uncertainty by scaling the noise in one pixel, the one closest to the source. The scaling factor, NP, reflects the central PRF pixel's contribution to the total PRF volume. (The "PRF", or Point Response Function, array stores the PSF after blocking by the instrument pixels including any sub-pixel efficiencies.) The scaling uses only

one PRF, the centered one, i.e. the one for a hit at the center of a pixel. So this noise is equivalent to fitting a single central pixel with a fixed PRF.

For $\sigma_{centerpixel}$, it uses the output of Apex's *gaussnoise* by default, i.e. an estimate of background fluctuations. This does not take into account photon noise from the source itself. So it can only work in the background-limited case.

The estimate fails when pixels are large relative to the PSF width (e.g. IRAC1), being too low (see Fig. 1). It uses only the centered PRF, and ignores the effects of position uncertainty. This was the motivation for a new option for SNR.

2. A new SNR: Photerr

The goal was to find a quantity which could roughly approximate the scatter of repeated flux measurements in a "quick-and-dirty" way. Simulations of Spitzer IRAC1 point sources were made. The best estimate found was basically an aperture uncertainty, with the aperture (a box here) big enough to cover the core of the PSF.

We added the ability for the user to optionally put in total source photon noise explicitly through a gain factor. This brings it more in line with noise estimates from other programs, e.g. Sextractor (Bertin & Arnouts 1996).

In the full case (with gain), the new Photerr SNR "noise" is:

$$\sigma^2 = \sum_i \sigma_i^2 + F/g \tag{2}$$

where

$$\begin{aligned} \sigma_i &= \text{pixel uncertainties (tile pixels in multiframe)} \\ F &= \text{source flux (data units)} \\ g &= \text{gain } (e^-/\text{data unit}) \end{aligned}$$

The final σ is converted to flux units. The sum is over a box covering the core of the PRF, out to 10 % of the peak value of the centered PRF. This was estimated from the simulation and is currently fixed. The box is whole pixels; no sub-pixel summing is done.

If run without the gain, the σ_i should generally be data uncertainties (if reliable). This will include (most of) the noise from the source. In this case it is just an aperture uncertainty over the summing box.

If run with the gain, one should use the output of *gaussnoise* for the σ_i. This combines background noise + total source photon noise, which is reasonable. But keep in mind the Apex fitting incorporates the data uncertainties if the user provides them, so this will work best when the data uncertainties match the background fluctuations + gain model (as they mostly do in the simulated data of Fig. 1 – note the similarity of the top two estimates).

Eq. 1 with data uncertainties and without the gain is the current recommendation. It matches the variance of flux measurements of the IRAC1 simulated data about as well as other estimates (Fig. 1), and the user doesn't need to know the gain.

An additional test on real data was done on a single mosaic of SWIRE MIPS 24 μm data (Spitzer AOR 5880832 downloaded from the SSC archive). Using the data uncertainties is the best choice (slightly conservative), see Fig. 2. Note in this case how the data uncertainties (provided with the mosaic) are considerably higher than the estimated background fluctuations.

3. Notes

- Uncertainties in background level are not included.
- Uncertainties due to confusion are not included.
- For a stack of images (Apex), the Photerr SNR calculation does a single scaling from data to mosaic pixel scales. So there should not be too much variation in pixel scale due to distortion in the original data. If there is, use a comparable aperture uncertainty obtained from running *aperture*.
- There is no PRF-weighting in Photerr. In the specific case of Spitzer's relatively big pixels, we don't always know reliably which (sub-pixel) PRF to use.
- Like an aperture uncertainty, the Photerr (without gain) can, if desired, be interpreted as the error one would get if one moved a flat-topped PRF the width of the summing box over each pixel in the summing box, fit the central pixel only, and then took the uncertainty in the average of those fluxes.
- The Photerr SNR is only a working measure and has no formal statistical basis. Best estimates of true uncertainties come from repeated measurements and/or simulations of the data.

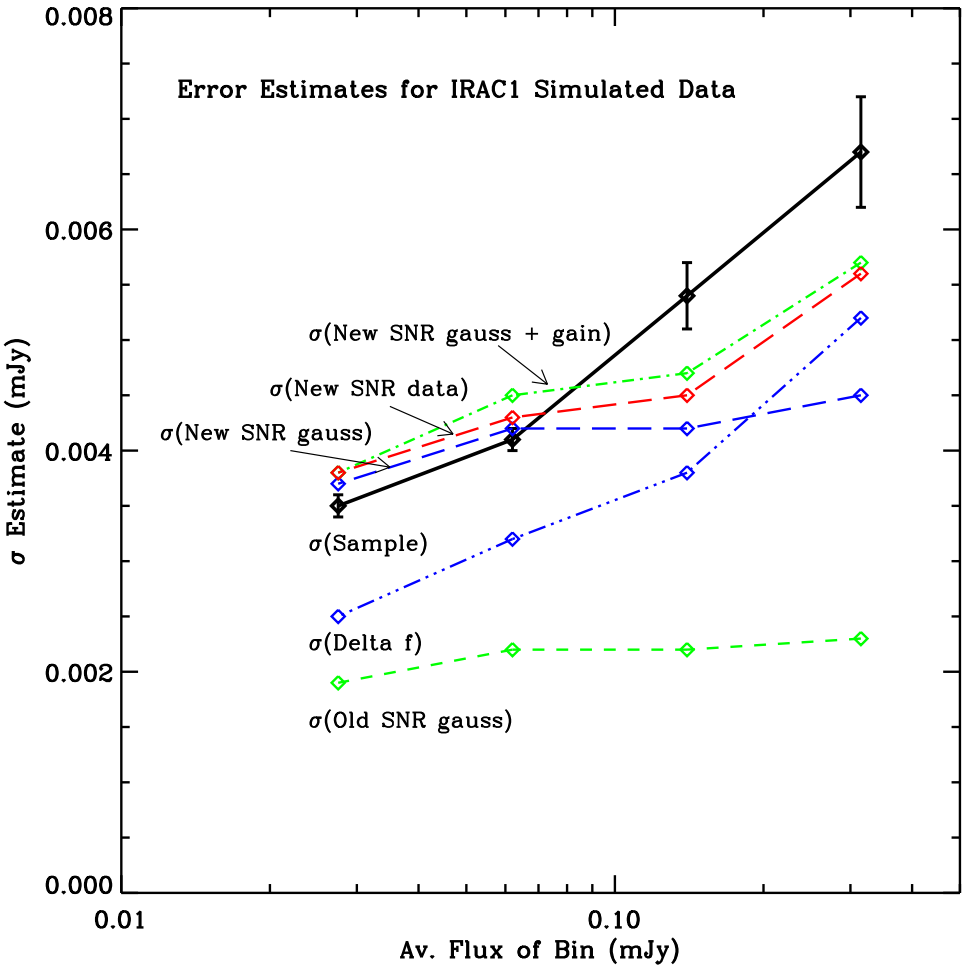

Fig. 1.— Flux noise estimates from IRAC1 (3.6 μm) simulated data. The "true" noise is taken to be the gaussian variance of measurements about their true values (black, solid). Binning a number of objects by flux is used to simulate repeated measures. Error bars are based on Poisson statistics in each bin. Apex quantities shown are 1) old SNR with gaussian background noise (green, short dash); 2) Delta_flux (blue, dot-dot-dot-dash); 3) new SNR with gaussian noise, but no gain-corrected source photon noise (blue, long dash); 4) new SNR with gaussian noise, with gain-corrected source photon noise (green, dot-dash); and 5) new SNR with data uncertainties (red, long dash).

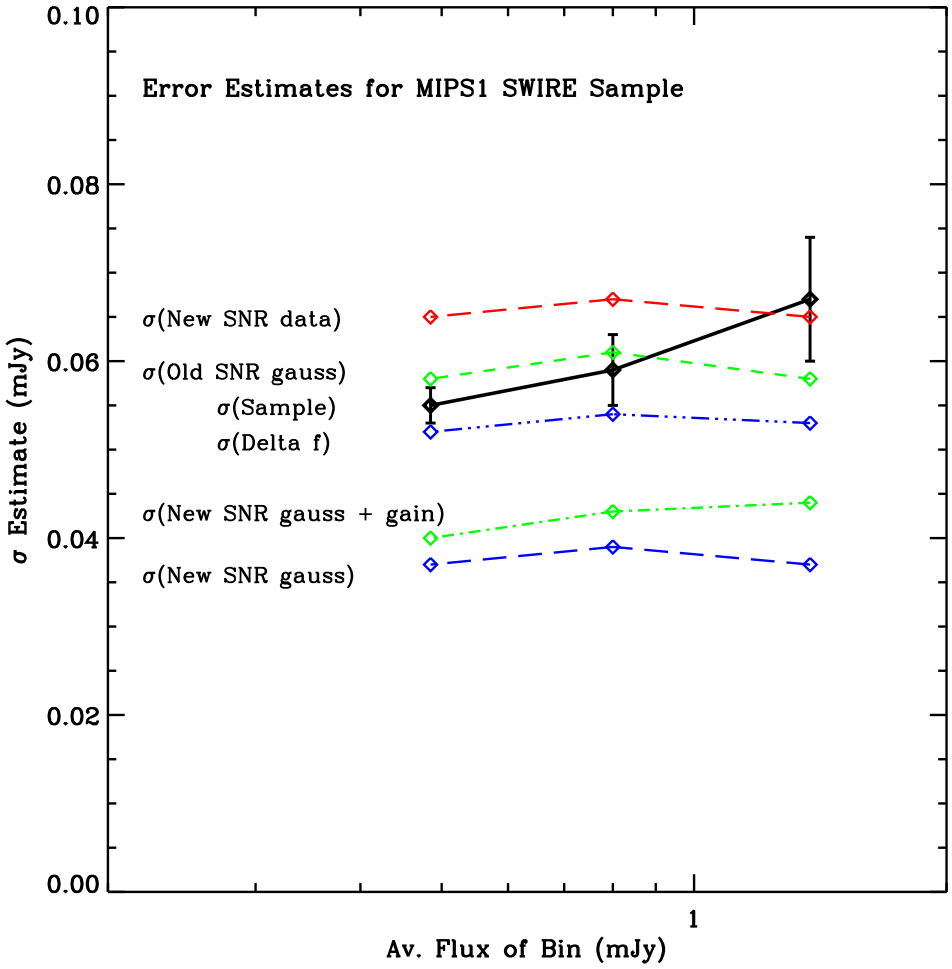

Fig. 2.— As in Fig. 1 with MIPS 24 μm SWIRE data from Spitzer AOR 5880832. True fluxes of these objects, mostly galaxies, are taken from the SWIRE catalog.

4. Instructions for Apex Users

The overall namelist switch, use_data_unc_for_fitted_snr = 1, adopts data uncertainties for the SNR if available; the default (= 0) is to use *gaussnoise* values. The recommended usage of this parameter is given below.

There are 2 new optional parameters available in the *sourcestimate* block. One is a switch, the other is the gain in e^-/data unit. Here's an example of what to add to the *sourcestimate* block of the namelist:

#Calculate SNR using photerr (0=no; 1=yes, no gain; 2=yes, with gain),

#Gain_for_SNR in e^-/data units, e.g. e^-/(MJy/sr),

Use_Photerr_for_SNR = 1,

Gain_for_SNR = 315.0,

The gain is that for an individual frame g(i). If you are working on a mosaic with Apex_1frame, you can input the Coverage Map and it will use the effective gain of an average, i.e. $N \times g$(i). Similarly for Apex on a stack, though in this case the user does not supply the coverages; it will look for the usual file where it stores the coverages for each tile. In the case of integration-time weighting with Apex, the input gain should be that for unit time.

Bad uncertainty values: In the case of bad uncertainty values, e.g. NaN's, in the summing area:

Any bad $\sigma_i^2 \Longrightarrow$ "Bad" SNR = -9.99

Typically -9.99 rounds off to -10.0 in the extract_raw.tbl. Often a user will select by SNR, so these sources will be lost in the final extract.tbl.

"Estimated" uncertainties: Data uncertainties could be estimated uncertainties, those calculated from the user-provided gain and read noise in the *snestimator* block, if that switch is on and data uncertainties are not provided. This is unchanged from previous versions.

Summary of normal usage:

1) Old SNR

use_data_unc_for_fitted_SNR = 0

Use_Photerr_for_SNR = 0, (not required)

2) New photerr SNR, with background noise, knowing the gain.

use_data_unc_for_fitted_SNR = 0

Use_Photerr_for_SNR = 2,

Gain_for_SNR = 315.0,

The gain above, in e^-/(MJy/sr), is approximate for IRAC1, 12 sec, 1 frame. From header: GAIN(3.3 e^-/DN) × EXPTIME(10.4 s)/FLUXCONV(0.109 MJy/sr per DN/s).

3) New photerr SNR with data uncertainties, no gain.

use_data_unc_for_fitted_SNR = 1

Use_Photerr_for_SNR = 1,

Bertin, E. & Arnouts, S. 1996, *Astr. Astrophys. Suppl. Ser.* **117**, 393-404.

Bevington, P.R., 1969, *Data Reduction and Error Analysis for the Physical Sciences* (New York, NY: McGraw Hill).