Specification Document: Revision to the MLE algorithm for CSC source fitting

Rafael Martínez-Galarza Date: January 25, 2024

1 Motivation: QA triggers in MLE algorithm

The maximum likelihood estimation (MLE) method was designed for the Chandra Source Catalog 2.0 in order to better estimate the position, the extent, and the likelihood of a given detection being a true detection. The algorithm was inspired by the `emldetet` algorithm from XMM Newton. The idea is simple: given a distribution of source and background counts within the detection region of a CSC source in a single observation or on a stack of observations, can we maximize the likelihood that the distribution of counts is being produced by an elliptical model convolved with the Chandra PSF at that location?

The maximization of the likelihood is done by fitting the convolved elliptical model to the data and then by minimizing a statistic appropriate in the Poisson regime, namely the `cstat` statistic, using a particular optimization method. The optimization method used in CSC processing is Sherpa's `moncar` method, which is an adaptation of the Differential Evolution algorithm of Storn & Price, 1997. This is basically an evolutionary algorithm in which each possible solution to the optimization problem (i.e., a set of parameters for the Gaussian model) is evolved according to certain rules, until a fitness function (given by the statistic) is is maximized.

The MLE method is applied to bundles (a collection of detect regions that overlap), and in fact involves two fits for each member of the bundle: a first fit in which the detect region is expanded by a factor of two, and a second fit where the detect region has the original size, but has been re-centerwd to the best-fit position of the first fit. Of relevance is the fact that both of those fits are done in an iterative way for the bundle, i.e., only one source in the bundle is fitted at once, starting with the brightest source, with the results of previous fits added as fixed parameters to subsequent fix.

This approach has resulted in a large set of satisfactory fits for CSC 2.0 sources, but also in a large number of triggers (what fraction of the total?) for the manual Quality Assurance (QA) interface. QA triggers occur when there is reasonable doubt that the fit is imperfect. Specifically, triggers occur in the following cases:

- **BundleOverlap**. This QA is triggered if the best-fit location of a source falls within 110% of the detect region for another source, i.e., if two detections have fitted positions that are too close to each other.
- **MefFitOutside** This QA is triggered if the fitted position falls outside of the expanded (2X) notice region. Note that the fit is done only noticing the counts within the detect region for purposes of the statistic calculation.
- **MrfFitOutside** This QA is triggered if the fitted position in the second fit falls within the 70% limit of the re-positioned, unexpanded notice region
- **HighEllipticity** This QA is triggered if the fitted elliptical source model has an ellipticity higher than 0.9.

Of these four types of triggers, the BundleOverlap QA is the one that gets triggered more often, about 50% of all QA triggers, followed by the MrfFitOutside (34%), MeFFitOutside (20%), and HighEllipticity (20%).

The goal of this specification document is to provide a set of recommendations for modifications to the MLE algorithm that would results in less manual QAs being triggered during automatic processing of the catalog properties. This amounts to finding alternative fitting procedures that are able to perform a meaningful automatic fit, even if the four triggers described above take place.

2 Finding the root of the problem

The QA statistics described above appear to indicate that source crowding is responsible for a large fraction of QAs triggered. In fact, most of the BundleOverlap triggers take place in regions that are crowded with detections that are very close to each other, such as in the region neighboring the galactic center, SgrA*. In such cases, source region overlapping is common, and it could be difficult for an optimizer or MCMC sampler to decide which counts correspond to a given source, *unless a simultaneous fit approach is employed*. This is because the sequential fitting currently employed is specifically told to fit a single source within a given notice region, but in such regions more than one source can be present within a given notice region.

But there is more to it than just source crowding. It often occurs, as the MefFitOutside and MefFitOutside triggers indicate, that the distribution of counts are often compatible with the source being located *outside* of the notice region. This can happen, for example, if the original position passed to the algorithm as an initial guess, which comes from the wavelet-based `wavdetect` algorithm has missed the true location of the source, producing an original an initial detect region that does not contain the peak of the emission. But this situation can also occur if bright nearby source have some of their PSF spilled into the notice region of interest.

Finally, if two or a few sources are too close together, even if regions that are not very crowded generally speaking, the notice region of the Sherpa fit in the current pipeline can include pixels whose total flux is contributed from more than one source, and this can lead to highly elliptical extended source regions. Here again, simultaneous fitting of the sources together is likely the best approach.

In general, I have identified three types of QAs that relate in one or several ways to the causes exposed above, and to the specific trigers:

- **Case 1**: A single wavdetect detection, but notice region too small (ECF90 region). This can trigger a MRF or a High Ellipticity QAs. See Figure 1.
- **Case 2**: Two or a few detection very close to each other, so that their regions overlap. This can triger MEF, MRF, Bundle Overlap, and Ellipticity QAs. See Figure 2.
- **Case 3**: More than a few sources closely overlap each other. This case produces mostly Bundle Overlap QA triggers, but also MRF and Ellipticity QAs in smaller numbers. See Figure 3.

One other factor that contributes to QA triggering and that ultimately reflects in the values reported in the CSC database is low-count sources. Basically it is very hard for an optimizer or an MCMC sampler to converge when only a few counts are available to perform the fit. Very rarely these very low count observations are selected as those having the highest likelihood, and this can happen for example if a particular low-count source has a background with exactly zero background counts. In that case, the low-count source in a given band and obsids will have a likelihood higher than other obsids and bands, even if the source has more net counts in other bands. Maximum likelihood should therefore not be the criterion to choose the nominal astrometric properties in the catalogs. In Fig. 4 I show that significance relates better to the net source counts than the likelihood.

2.1 A possible approach

Currently, the MLE algorithm is applied to Chandra Source Catalog data in a per-band and per-obsid basis. In the current approach, two fits are performed for each source: first, a fit is performed using an expanded version of the original wavdetect region as the notice region. After this first fit is performed, a second fit is done after the notice region is re-centered to the position of the first fit, and reduced to be the ECF90 region corresponding to the PSF at that location. Additionally, another fit is performed with an extended source model. If the fit with the smallest fit statistic is the extended model, then the source is assumed to be extended. The fit parameters passed to populate the database for a source detected in several obsids, correspond to the fit in which the likelihood ends up being higher. Again, using the likelihood instead of the signal-to-noise (significance) is not optimal, because of reasons discussed above related to background counts. Finally, sources that belong to the same bundle (their detect regions overlap) are fitted sequencially, fitting the brightest source first, then freezing the parameters of that source as part of the background to fit the second brightest source, etc., until all sources have been fitted.

With the approach described above, a total of over 15,000 QAs were triggered during CSC2.0 processing (with the number decreasing in further editions as the original ones were corrected). In order to achieve our processing goals and timeline for CSC2.1 and subsequent versions, we need to reduce the number of QAs by at least 50%. In Table 2.1 below I show the fraction of the different types of QAs triggered during CSC2.0 processing for the different editions. Note that the Bundle Overlap QA trigger accounts for about half of all the QA. Solving that type of QA is therefore a main goal of this specification document. Other type of QA that significantly contributes is MRF QAs.

	Edition 1	**Edition 2**	**Edition 3**	**Edition 4**	**Edition 5**
MRF (%)	34	20	16	12	9
MEF (%)	20	23	28	25	28
Bundle Overl. (%)	**51**	**32**	**30**	**34**	**44**
Ellipticity (%)	20	20	24	28	32
Src Ext. (%)	8	10	9	9	19
TOTAL (number)	15608	7292	2541	1021	474
Man rejected (%)	14	6	2	0	1

No single approach will equally reduce all types of QAs for all detections. Ideally, one would want to simultaneously, rather than sequencially, fit sources that are clustered together using a single model that assumes multiple sources. But this becomes computationally intractable for more than a few sources, especially if they are detected in many obsids, as is the case for the most popular regions of the CSC, such as the galactic center. Therefore, we need to come up with an approach that a) reduces significantly the number of QAs, and b) differentiates between cases in order to obtain the optimal reduction of QAs in each case. Ideally, in order to increase efficiency, we would also prefer not having to perform the expand-fit-recenter-refit approach described above. Finally, while more state-of-the-art algorithms for MCMC optimization and sampling are currently available (such as pymc3), we prefer for now not to modify the core fitting procedures that use Sherpa.

Below, I propose a novel approach to perform MLE fitting.

2.1.1 A new roadmap for MLE fitting

I recommend to perform the MLE fitting with different approaches depending on the clustering properties in the vicinity of each detection:

1. If the bundle has only one source: Perform a SINGLE Sherpa fit for each obsid/band, using the detect region (expanded by 50%) instead of the ECF90 region as the notice region. This expansion is needed because sometimes the original detect region misses the pixels where most of the counts are. Allowing for a larger notice region will correct

2. If the bundle has between 2 and 5 sources: perform a simultaneous fit **over sources**. Use a common, large notice region that encompasses all detections in the bundle, and a common background region for them all. We specify how to select the regions below.

3. If bundle has more than 5 sources, leave pipeline unchanged, i.e., perform expand-fit-recenter-refit in a sequential fashion for all sources in the bundle. Some QAs will still be triggered, but that is the price to pay if we do not want a major overhaul of the MLE pipeline.

4. The likelihood of each detection when a simultaneous fit is performed can be calculated by setting the amplitude of the source of interest to zero, after the fit has been performed, and then evaluating the statistic of the model with those parameters. This provides a statistic for the null hypothesis of no source being at that location, while the statistic of the full fit corresponds to the hypothesis of a source being present. A likelihood ratio evaluation can then be performed, just as currently set in the pipeline, to reject or accept the null hypothesis. This is done for each of the detections that were fitted simultaneously.

5. For simultaneous fits, two fits should be performed: the first fit uses a point source model for all sources in the bundle, while the second fit is performed using an extended source model for all

sources, with their positions frozen to the values of the first fit. MCMC draws for the extended source model should only be attempted if for at least one of the sources, the upper limit of the deconvolved source extent is larger than the resolution element (1 pixel) at the resolution in which the fit was performed.

6. The determination of whether a source is extended or point-like is done based on the same premises and set of criteria as was done in CSC 2.0 processing. The difference is that the likelihoods used in the case of simultaneous fits should correspond to the modified likelihood calculation that is described earlier. The calculation of the source counts should still be done using the ECF90 regions, even though we have now used the FSRCREG regions (expanded or not) as the notice region for the fit.

2.1.2 Simultaneous fitting

In general, it is important to make a distinction, when we refer to a simultaneous fit, between a simultaneous fit because there are multiple sources in a bundle, and a simultaneous fit because there are multiple observations in which the bundle appears. In the context of the present specification, when we refer to a simultaneous fit, we refer to the first case, i.e., a simultaneous fit over sources (i.e., we consider a model that has more parameters, one set of position, amplitude, and extent parameters for each source in the bundle). The CSC 2.0 production version of the pipeline already accounted for simultaneous fits *across observations*, and we are not proposing a change regarding that here.

In order to perform a simultaneous fit of up to 5 sources in the bundle, it is necessary to define a single notice region that encompasses all sources in the bundle. Only pixels within the notice region are used to calculate the fit statistic in each iteration. It is therefore important that this region includes the emission from all sources in the bundle. In order to define this region, one can use the union of all the individual detect regions in the bundle. In sherpa, it is easy to combine regions. For example, if a bundle has 3 sources, the corresponding DS9 regions `src1.reg`, `src2.reg`, and `src1.reg`, can be added in sherpa:

```
ui.notice2d('src1.reg'+'src2.reg'+'src3.reg')
```

Note that the source regions 1, 2, and 3 here correspond to the STACK detection region files of the sources in the bundle. They do not refer to the observation (obsid). If the bundle appears in more than one observation, then the regions that are added (one for each of the three sources) are the same for all obsids, as they correspond to the stack detection.

In a similar fashion, the background region for the fit can be set to the union of the individual background regions of the sources (minus any overlapping sources that are not part of the bundle). As usual, the background is fitted first, and then frozen for the subsequent fit of the sources. Then the model is defined as the sum of individual `sigma2d` sources and the common background. So, for a single observation (observation 1), a model including three sources (A, B, C) looks like:

```
sA = ui.sigmagauss2d('simA')
sB = ui.sigmagauss2d('simB')
sC = ui.sigmagauss2d('simC')
# Convolve with PSF, add bckground and exposure map
ui.set_source(1, b1 + (psf1A(simA)+psf1B(simB)+psfC(simC))*e1)
```

Where b1 and e1 are respectively the background map, and the normalized exposure map for the observation in question, and psf1A, psf1B, psf1C, are the PSFs at the locations of each source.

If more than one observation contributes to the set data being fitted for that source, then the model should be defined for each of the contributing datasts. So, if there was a second observation covering the same bundle, the model for that observation would look like:

```
sA = ui.sigmagauss2d('simA')
sB = ui.sigmagauss2d('simB')
sC = ui.sigmagauss2d('simC')
# Convolve with PSF, add bckground and exposure map
ui.set_source(2, b2 + (psf2A(simA)+psf2B(simB)+p2sfC(simC))*e2)
```

Where the index '2' now referes to the secon observations, so the PSFs, and the exposure and background maps now correspond to the second observation.

For this two obsids, the call to the simultaneous fit function should be:

```
ui.fit(1,2)
```

2.1.3 Likelihood estimation for simultaneous fits

In the CSC 2.0 production pipeline, likelihood estimation in MLE is done in terms of a hypothesis rejection test. We compare the statistic of the null hypothesis fit (the background-only model) with the statistic of the fit including the source. The likelihood is estimated as the result of the `igam` function (which is the functional form that best describes the distribution of the statistic assuming the null hypothesis) evaluated for the differences between the two statistics (`delta_stat`). If the resulting statistic difference is very far off in the wing of the distribution, then the null hypothesis (no source) in easily rejected. This was done in a source-by-source basis, as a fit always was done for *single sources*.

For the case of the simultaneous fit this approach needs to be modified. The reason is that since we now have a single statistic for all n sources in the bundle during a simultaneous fit, we can no longer derive a single statistic for each source.

An alternative for estimating the likelihood in this case is as follows: instead of using the statistic of the background fit as the null hypothesis statistic (`cnull`), we can use the statistic estimated by setting the amplitude of the source in question to zero (once the full fit has been performed), leaving all other parameters at their best-fit values. So, for one of the sources in the fit `delta_stat` is defined as before:

```
delta_stat = cstat - cnull
```

But in this case `cstat` is the final statistic of the full fit, including all sources, and `cnull` is the estimated statistic when the amplitude of the source in question is set to zero. The rest of the likelihood calculation remains the same. To estimate `cnull`, Sherpa's `calc_stat` tool can be used, after setting the parameter corresponding to the amplitude of the source in question to zero.

As for the `num_free_pars` parameter in the call to `igam(num_free_pars / 2.0, delta_cstat)`, it should be set to the total number of free parameters, which equals the total number of sources in the bundle times the free parameters for each source. In the more general case, this corresponds to 6 parameters per source: two coordinates, the amplitude of the Gaussian model, the semi-major and semi-minor axes of the extent ellipse, and the roll angle of the extent ellipse.

2.2 Fitting with a higher resolution PSF

Of relevance to the present work is the fact that we have implemented improvements to Sherpa 4.12 that allow us to evaluate a model in an arbitrary grid. This is of relevance for source fitting in MLE, because if a PSF is available at a higher resolution that the data resolution, we can perform the convolution in the high resolution grid prior to the calculation of the fit statistic. We have demonstrated (`https://bit.ly/35N8uEq`) that this improves the accuracy of the fit. In CSC2.0, the input to MLE (count images, PSF images, exposure maps, background images, etc.) are binned to guarantee sufficient sampling of the PSF, so that the PSF FWHM corresponds to between 3 and 6 image pixels.

In order to assess if additional rebinning of the PSF to even higher resolution would further improve the Sherpa fit within MLE, I have performed a fit of a source that triggered a QA, namely `qa_mrf_acisfJ1821102m155828_001_0037_N022`. This is a single source in a single obsid stack, which in band m has only a few counts in the source region. I performed the fit using a `BIN=1`, and a `BIN=1` version of the counts image, the background image, and exposure images, and then used both a `BIN=1`, and a `BIN=0.5` versions of the PSF image. Both fits resulted in satisfactory agreement with the data, but there are no significant differences between the best fit values in both cases. In fact, it seems that for the fit with `BIN=0.5`, the credible intervals for the source position are wider than in the case of `BIN=1.0`, as can be seen in Fig. 5. While this seems to contradict our initial assessment regarding the improvement of the fit with a finer PSF, it is perfectly possible that in a source with such a few number of counts, PSF resolution is not the determining factor in improving the accuracy.

If the b band image (which contains a higher number of events) and b band PSFs are used instead, the difference between `BIN=0.5` and `BIN=1.0` in terms of the fit results is minimal, as shown in Figure 6.

We show the best fits for each case in the b band in Figure 7. We note that in this particular case, a higher PSF resolution does not result in significant gains in accuracy. This might not be the case in more luminous events.

In terms of performance, optimizing the parameters using the *simplex* method takes about the same computing time for BIN=1.0 (0.24s) and BIN=0.5 (0.25s). As for the MCMC sampling with `get_draws()`, it takes 11.2s with BIN=1.0 against 15.27s with BIN=0.5. For catalog processing purposes, therefore, the impact on performance if BIN=0.5 is used is of the order of 50% additional computing time.

Given the fact that MLE already matches the image resolution to the PSF resolution that gives 3-6 pixels across the FWHM (this should be increased to 6-12 pixels starting in CSC 2.2), and given that many of the QA sources might be low count sources, I do not currently recommend increasing the PSF resolution further.

2.3 Selecting the right band and the right notice region

In MLE, the fits corresponding to the band with the highest derived likelihood is chosen to populate the database. A number of QA triggers happen because the band chosen has effectively zero counts within its background region. This results in a very large likelihood (i.e., the source is detected with close to absolute certainty in absence of background). The problem is that this to happen in bands where the source region itself does not contain many counts, either because it is faint, or because the exposure time was short. As a result, the count distribution in that chosen band is not the best representation of the source position, because the few counts are scattered over the source region. In Fig. 8 I show an example from an actual QA trigger, which is the same QA as before, namely `qa_mrf_acisfJ1821102m155828_001_0037_N022`. The QA is of the MrfFitOutside type, which means the MRF fitted position falls too close or outside of the edge of the source region. The chosen band (m) has a high detection significance (due to the fact that background is effectively zero), but the counts in the source region are too few to determine the position accurately, and as a result, the fitted position ends up outside of the 70% of the notice region of the MRF fit, triggering a QA.

I am able to reproduce this QA in Sherpa. The QA can be resolved if a different band is used, for example the b band, which has a lower likelihood but it is a fairer representation of the source flux distribution. But it can also be resolved if the fit is performed withing the source region instead of the ECF90 region, which is the default in MLE for point sources. This is because the ECF90 region cuts trough some of the pixels containing counts, and this affect the quality of the fit. It is advisable in these cases to relax the ECF90 constraint, and perform the fit in a larger area (e.g., the source region). This alone might resolve between 20 and 30% of the QAs, which are of the MrfFitOutside type.

In the b band the detection has a lower significance, because there are background counts present. But counts are significantly more numerous in the source region, as shown in the Figure 8. This makes the determination of the position much more reliable. In order to corroborate this, I fitted the b band using the same Sherpa model and limiting the notice region to the source region instead of the ECF90 region. The result, compared to the QA position of the fit is shown in Figure 9.

The problem with the QA fit is even more evident if we look at the MCMC traces from the fit. We show them for bot the original fit and the new, improved fit in Figure 10.

There are cases, such as acisfJ0522499p332815-001N021-b0227, where wavedect only finds one source when in fact there are too source close together. In this case, a QA is desirable, because fitting two sources with a single region that cuts through both sources is complex. Either we expand and do a simultaneous fit that might not be a bundle, or manually input the source positions, which is what QA does anyway.

Perhaps wee need to add a step to MLE in which the fit is redone if the fitted position is cloe to the edge of the SRC region, whih a region centered on the fitted position.

3 Algorithm

Below, the revised MLE algorithm is described. The algorithm is run per bundle. No need for re-bundling with respect to the bundling strategy adopted in CSC2.0 is needed.

3.1 Input to MLE

1. Images of the source
2. BKG table model
3. PSF images
4. ExposureExposure maps
5. Region files, containing src (ECF90REG and FSRCREG) and bkg (BKGREG) regions
6. Pixmask files
7. Config file

Note that for §3.3.2 below, a single set of region images (count images, exposure map, background map, and pixmask) should be used for the bundle. That is, there should *not* be a set of region images and maps for each source in the bundle. A single `ctsimg3.fits` image should be generated for the bundle prior to the analysis, regardless of how many sources there are in the bundle. The same applies for the other maps (background, exposure, pixel map).

This only applies to §3.3.2. below. For bundles with single sources, or with more than 5 sources, the same approach for the creation of the region images should be used ans in CSC2.0 processing.

3.2 Background fit

To determine the background level in the BKGREG region of a given source, the following Sherpa model should be used:

```
bkg = tablemodel.bkgtbl1 * tablemodel.emap1
```

Note that both factors involve table models. These are provided respectively by the background and normalized exposure maps generated by the pipeline The amplitude of this model is then fixed once fitted for subsequent fits.

For a given bundle, the background region used to perform this initial fit depends on the number of sources in the bundle:

- For bundles with a single source, the background region should be the single BKGREG for the source.
- For bundles having between 2 and 5 sources, the background region should be the union of the individual BKGREG regions.
- For bundles having more than 5 sources, the fit is done sequentially. For the brightest source, its background region (minus any overlapping sources) is used. For subsequent sources, the background consists of the BKGREG of the source in question, with the fitted models of previously fitted sources added as additional background.

3.3 Fit sources

The specific procedure for fitting will depend on the number of sources in the bundle. If the bundle has a single source, a fit using an expanded version of the FSRCREG will be used. If the bundle has between 2 and 5 sources, a simultaneous fit over sources, as well as over obsids, is performed. If the bundle has more than 5 sources, a sequential fit over sources is performed.

3.3.1 A single source in the bundle

1. If the src is has a crater flag set to True then the likelihood is set to infinite
2. Fit the src model (assuming the number of counts > 3):

```
src = (tablemodel.bkgtbl1 + psfmodel.psf1(sigmagauss2d.source1)) * tablemodel.emap1
```

   - Fit is to be performed in physical coords.
   - For a point src only the amplitude and the (x, y) positions are thawed.
   - The two sigmas modeling width are frozen at one detector pixel, the theta parameter is fixed at the default value of 0
   - For an extended src all six parameters are thawed: the two sigmas, (x, y) positions, theta and ampl
   - Initial starting position of src candidate is the result of sherpa's guess function evaluated on the detect FSRCREG region, expanded by 50%. Originally, MLE used the wavdetect position.
   - The simple bounds for the (x, y) positions can vary up to 4 and 8 pixels with respect to the starting position (runtime parameters) for the starting point and extended src, respectively. Note: I am not convinced that these bounds are necessary. I think the model should be allowed to search the entire region. Best is to keep it as optional.
   - The simple bounds for the amplitude (bound-spec): [8.3415176093e-05, 2.0e-2].
   - The Fit is to minimize the cstat using the moncar optimization algorithm
     - cstat is calculated using the data points within the FSRCREG expanded by 50%, for both point or extended models.
     - The position parameters can vary within the a fixed number of pixel from the starting position. This number of pixels should be a free parameter of the pipeline.
     - If fitted position is outside of the ECF90REG, send the source for QA
3. Calculate the edge valid flag.
   - Calculate the minimum distance to pixmask
     - `min_src_valid_distance`: calculate the minimum distance from fitted position to nearest valid pixmask region
     - `min_src_invalid_distance`: calculate the minimum distance from fitted position to nearest invalid pixmask region
     - `min_wavedetect_valid_distance`: calculate the minimum distance from wavedetect position to nearest valid pixmask region
     - `min_wavedetect_invalid_distance`: calculate the minimum distance from wavedetect position to nearest invalid pixmask region
     - If `min_src_invalid_distance < min_src_valid_distance` then `set min_src_distance = - min_src` else `min_src_distance = min_src_invalid_distance`
     - `wavedetect_src_distance = min_src_invalid_distance - min_wavedetect_invalid_distance`
     - If `min_src_distance < 0.0` set `edge_valid` to False
     - If `min_src_distance < npixel2pixmask` (a runtime option 0.25) and if `wavedetect_src_distance > 0.` set `edge_valid` to True else set `edge_valid` to False
     - If two previous conditions are not met then set `edge_valid` to True
4. Calculate the log-likelihood
   - Calculate cnull, the background model in the fit (FSRCREG) region
   - `delta_cstat = cstat of point/extended src - cnull`

- `P = igam( num_free_pars / 2.0, delta_cstat)` where igam is the incomplete gamma function
- The log-likelihood `L = -ln( P )`

5. If `edge_valid` is False, set `L = NaN`
6. Calculate the bkg, point and extended source model count
7. Decide if last fitted src is a point or extended. A source shall be labelled as extended if all of the following conditions are met (`revised_spec` and `initial_spec`):
   - `extended_model_count_<band>  >_f_<band> * 30`, where
     - `f_u = 0.027`
     - `f_s = 0.30`
     - `f_m = 0.33`
     - `f_h = 0.36`
     - `f_b = 1.0`
     - `f_w = 1.0`
   - `ext_mdlcnt_b > 30.0`
   - `extsrc_llklhd_<band> > 10`
   - extent size, ie `sqrt( ext_smaj_<band> * ext_smin_<band>) > 2 * block pixel size`
   - `Both ext_smaj_<band> and ext_smin_<band> >` greater than the lower limit of the parameter ranges
   - If any of items a-d fail, the source will be classified as a point source for that band
   - We should define a "possible extent" flag to be set if the source fails because of d or e (or both) only
8. Once the brightest source in the bundle is determined to be a point or extended source, then that source fit and the background fit are frozen and added to a new source model to fit the next brightest source in the bundle, like:

```
src = (tablemodel.bkgtbl1 + psfmodel.psf1(sigmagauss2d.src1)+
psfmodel.psf2(sigmagauss2d.src2)) * tablemodel.emap2
```

   Where src1, psf1 are the source model and PSF for the previously fit, brighter source, and src2, psf2 are the source model and PSF for the current source (i.e., next brightest src).
9. The errors on the fitted parameters are calculated using the draws from shepa's `get_draws` function. The algorithm may be used to explore parameter space a suspected minimum.
   - Approximate the diagonal covariance matrix by using sherpa's `int_unc`
   - Set the sampler option of scale to the runtime 'scale' parameter (default: 10.0 and 0.1 point and extended src)
   - `set_prior`
     - Instantiate 2 and 4 sherpa Box1D class for point and extended src
     - Set box limits to the free parameter limits
     - `Set_sample_opt('scale', scale)` for ECF/FSRCREG.:
       scale = 1.0
       scale = 0.1
     - Freeze all box parameters
     - `set_sampler_opt('originalscale', 8 * [True])`

- `set_sampler_opt('priorshape', priorshape)` for ECF/FSRCREG:
  1. priorshape = [False, False, False, False, True, True, False, False]
  2. priorshape = [False, False, True, True, True, True, False, False]
- `set_sampler_opt('defaultprior', defaultprior)` for ECF/FSRCREG:
  1. defaultprior = [True, True, True, True, False, False, True, True]
  2. defaultprior = [True, True, False, False, False, False, True, True]

10. Repeat steps 1-11 for the remaining sources in the bundle

11. The bkg model fit and steps 1-12 are to be run for the individual obsid and all obsids (by simultaneous fit)

3.3.2 A bundle with between 2 and 5 sources

1. If any of the srcs in the bundle has a crater flag set to True then the likelihood for that source is set to infinite

2. Fit a simultaneous src model including all sources for which the number of counts larger than 3. So, for example, if there are 3 sources with more than 3 counts:

```
(tablemodel.bkgtbl1 + (psf1(sigmagauss2d.source1) +
psf2(sigmagauss2d.source2)+ psf3(sigmagauss2d.source3)) * tablemodel.emap1
```

   - Fit is to be performed in physical coords.
   - For a point src only the amplitude and the (x, y) positions are thawed for all sources.
   - The two sigmas modeling width for each source are frozen at one detector pixel, the theta parameter is fixed at the default value of 0.
   - For an extended src all six parameters for each source are thawed: the two sigmas, (x, y) positions, theta and ampl
   - Initial starting position of src candidate is the result of sherpa's guess function evaluated on the detect region formed by the union of the individual FSRCREG regions.
   - The simple bounds for the (x, y) positions can vary up to 4 and 8 pixels with respect to the starting position (runtime parameters) for the starting point and extended src, respectively. Note: I am not convinced that these bounds are necessary. I think the model should be allowed to search the entire region. Best is to keep it as optional.
   - The simple bounds for the amplitude (`bound_spec): [8.3415176093e-05, 2.0e-2]`.
   - The Fit is to minimize the cstat using the moncar optimization algorithm
     - cstat is calculated using the data points within the union of the individual FSRCREG regions, e.g.:
       `ui.notice2d('src1.reg'`'src2.reg'+'src3.reg')+
     - The position parameters can vary within the a certain number of pixels from the starting position. This number of pixels should be a free paramter.
     - If fitted position is outside of the resulting union of FSRCREG regions for any of the sources, send the bundle for QA.

3. Calculate the edge valid flag.

   - Calculate the minimum distance to pixmask
     - `min_src_valid_distance`: calculate the minimum distance from fitted position to nearest valid pixmask region

- `min_src_invalid_distance`: calculate the minimum distance from fitted position to nearest invalid pixmask region
- `min_wavedetect_valid_distance`: calculate the minimum distance from wavedetect position to nearest valid pixmask region
- `min_wavedetect_invalid_distance`: calculate the minimum distance from wavedetect position to nearest invalid pixmask region
- If `min_src_invalid_distance < min_src_valid_distance` then `set min_src_distance = - min_src_` else `min_src_distance = min_src_invalid_distance`
- `wavedetect_src_distance = min_src_invalid_distance - min_wavedetect_invalid_distance`
- If `min_src_distance < 0.0` set `edge_valid` to False
- If `min_src_distance < npixel2pixmask` (a runtime option 0.25) and if `wavedetect_src_distance > 0.` set `edge_valid` to True else set `edge_valid` to False
- If two previous conditions are not met then set `edge_valid` to True

4. Calculate the log-likelihood
   - Calculate cnull, the background model in the fit (FSRCREG) region
   - `delta_cstat = cstat of point/extended src - cnull`
   - `P = igam( num_free_pars / 2.0, delta_cstat)` where igam is the incomplete gamma function
   - The log-likelihood `L = -ln( P )`
5. If `edge_valid` is False, set `L = NaN`
6. Calculate the bkg, point and extended source model count
7. Decide if last fitted src is a point or extended. A source shall be labelled as extended if all of the following conditions are met (`revised_spec` and `initial_spec`): (Note. These conditions seem quite stringent. Can we do this just in terms of the size and the likelihoods, i.e., without the counts?)
   - `extended_model_count_<band> >_f_<band> * 30`, where
     - `f_u = 0.027`
     - `f_s = 0.30`
     - `f_m = 0.33`
     - `f_h = 0.36`
     - `f_b = 1.0`
     - `f_w = 1.0`
   - `ext_mdlcnt_b > 30.0`
   - `extsrc_llklhd_<band> > 10`
   - extent size, ie `sqrt( ext_smaj_<band> * ext_smin_<band>) > 2 * block pixel size`
   - `Both ext_smaj_<band> and ext_smin_<band> >` greater than the lower limit of the parameter ranges
   - If any of items a-d fail, the source will be classified as a point source for that band
   - We should define a "possible extent" flag to be set if the source fails because of d or e (or both) only

List of Figures

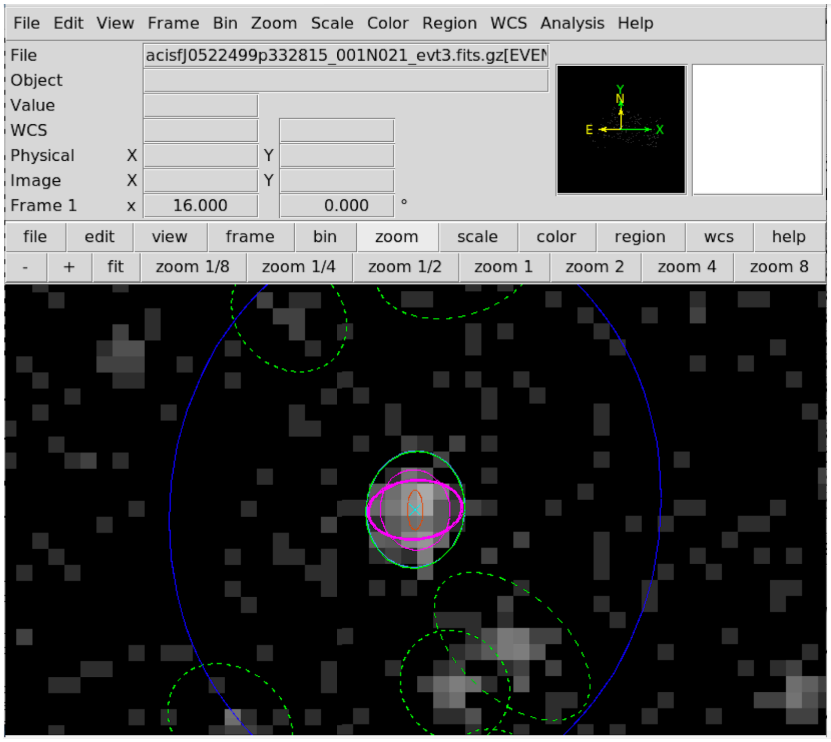

Figure 1: A notice region is too small.

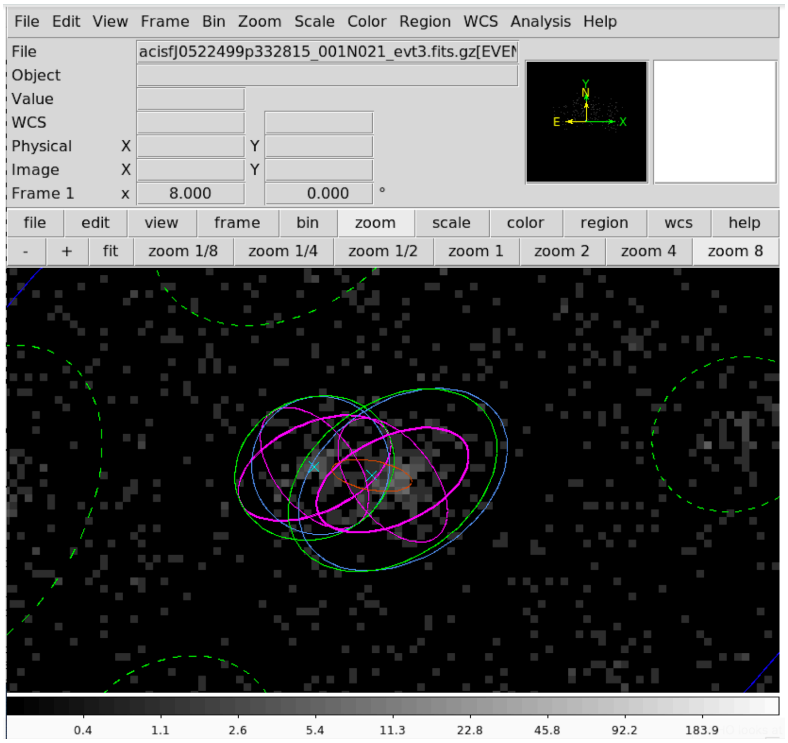

Figure 2: Two detections are too close together, and there is contribution from one source inside the other source's notice region.

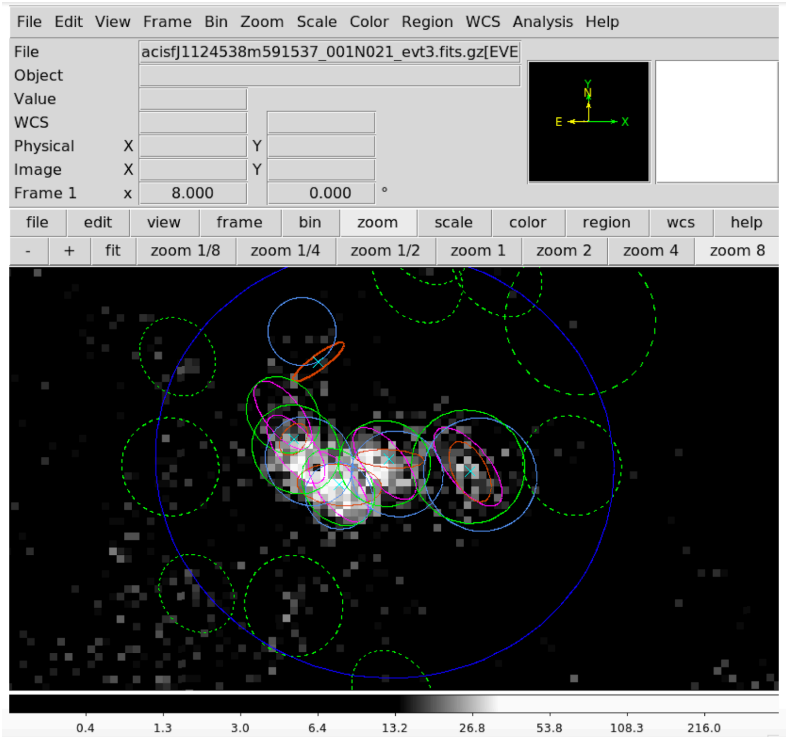

Figure 3: More than a few detections overlap significantly.

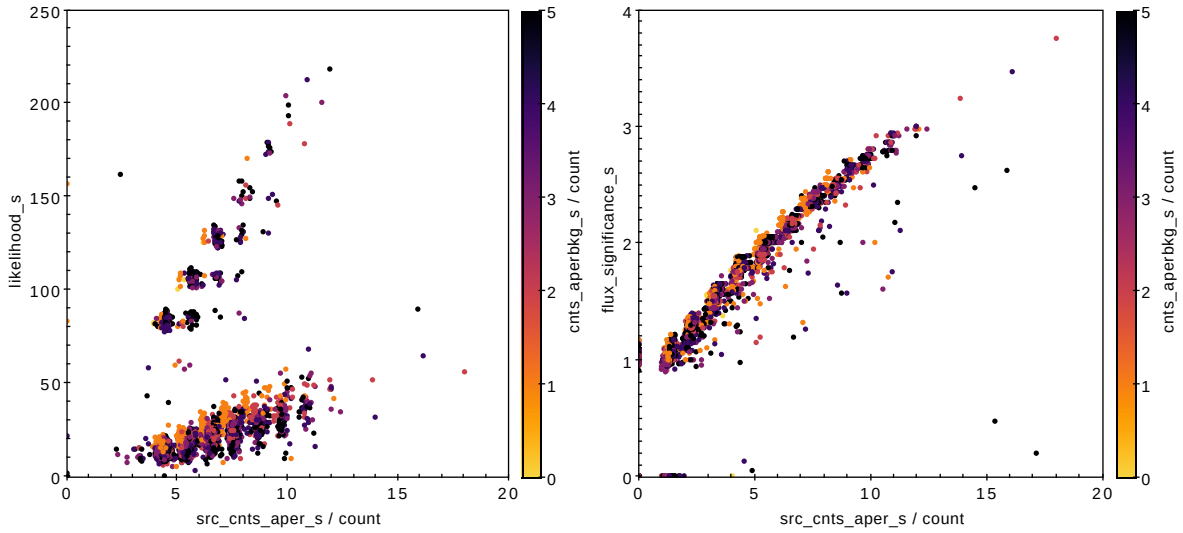

Figure 4: The correlation between the estimated source counts and likelihood (left) and significance (right), for the s band, and for CSC sources with less than 20 source counts. Note that the correlation is better defined for the significance. In addition, note that there is a group of zero background sources that are assigned a higher likelihood, despite them also having a very low number of counts.

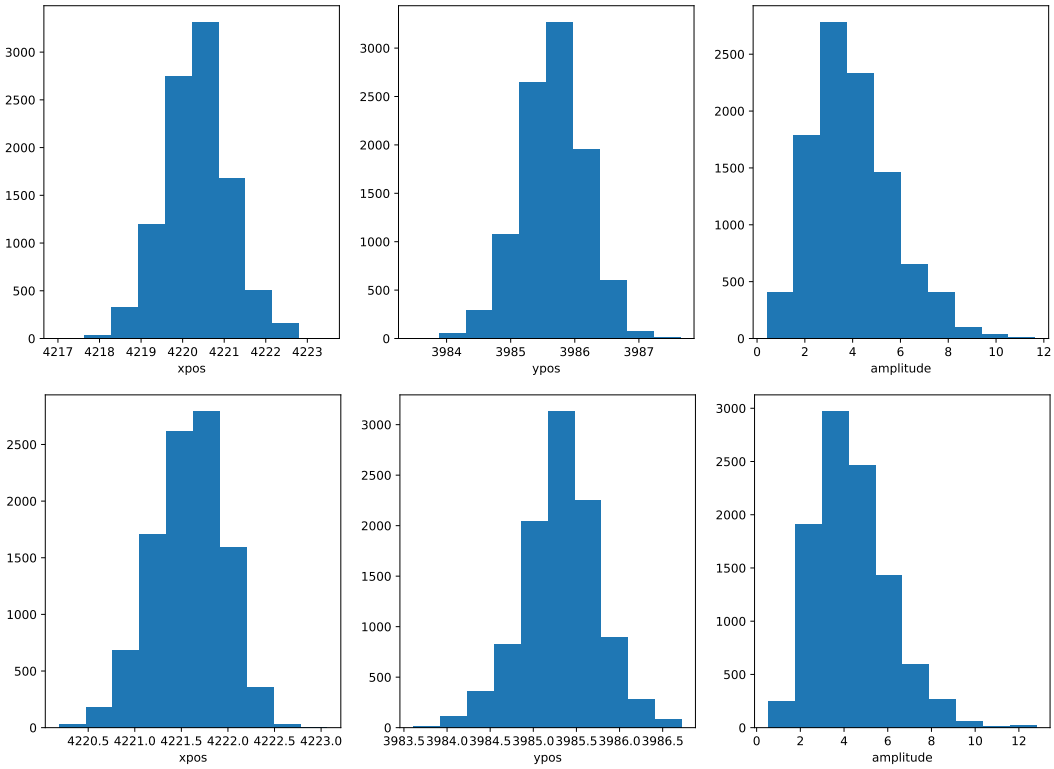

Figure 5: The PDFs for source positions and amplitudes, as derived from Sherpa's get draws. Both fits are for a single source in band m, which has only a few counts within the source region. The upper panel corresponds to a BIN=1 PSF resolution, whereas the lower panel corresponds to BIN=0.5.

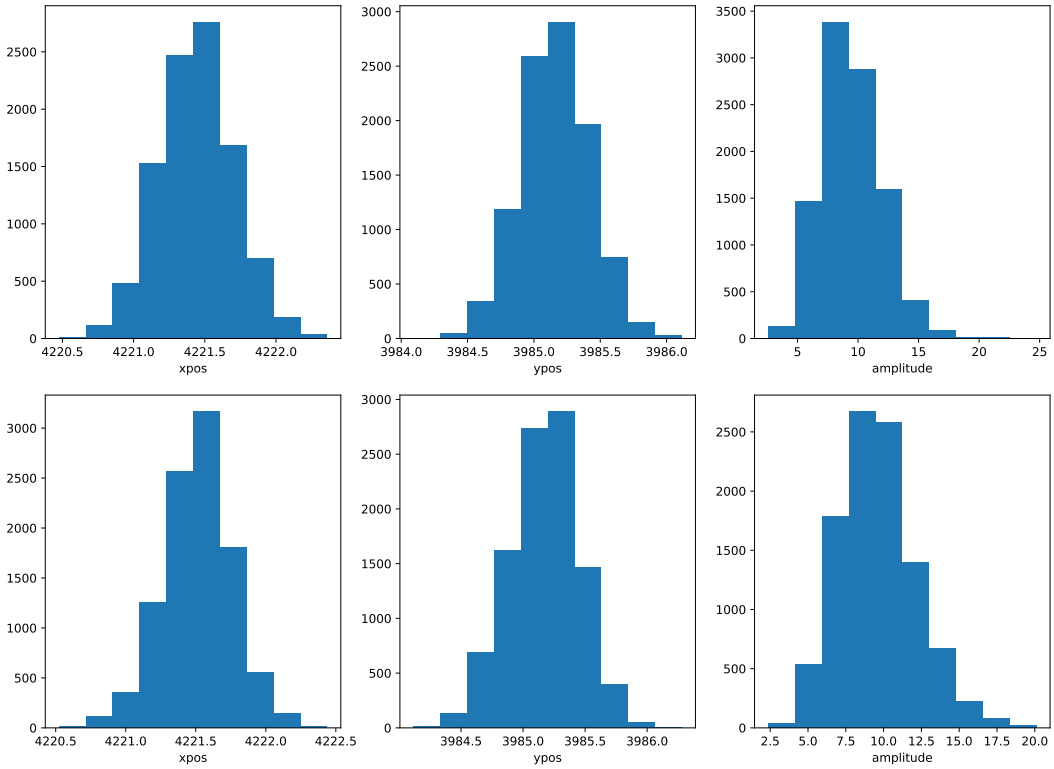

Figure 6: The PDFs for source positions and amplitudes, as derived from Sherpa's get draws. Both fits are for a single source in band b, which has only a few counts within the source region. The upper panel corresponds to a BIN=1 PSF resolution, whereas the lower panel corresponds to BIN=0.5.

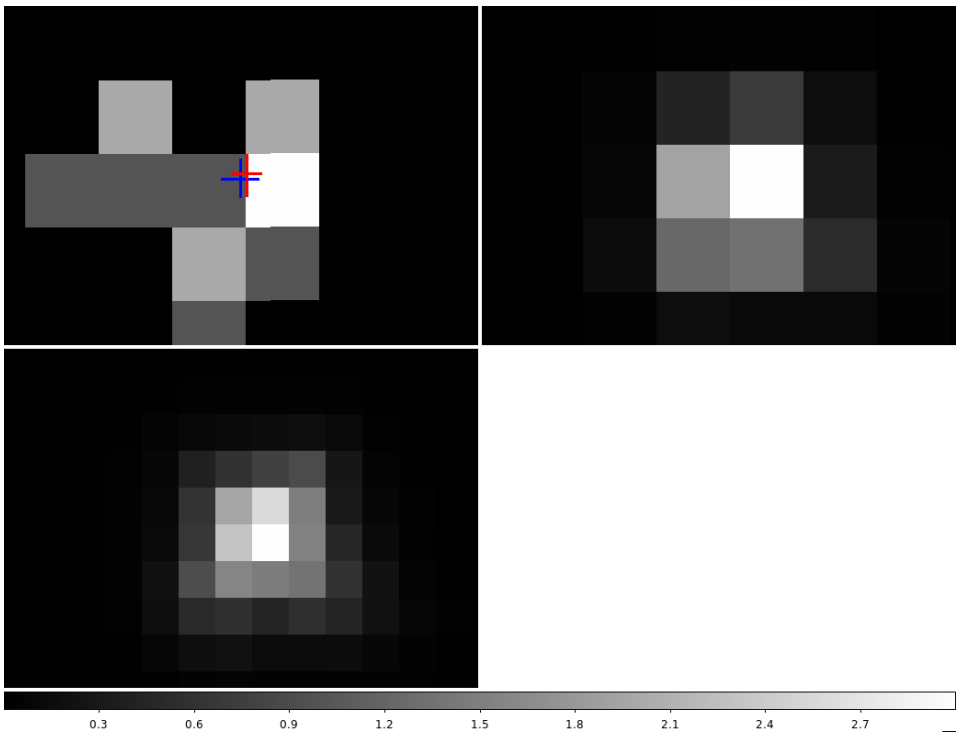

Figure 7: The best fit position and errors for acisfJ1821102m155828-001-0037 using the b band. The red bars correspond to BIN=1, whereas the blue bars correspond to BIN=0.5. Also shown are the PSFs at different resolutions.

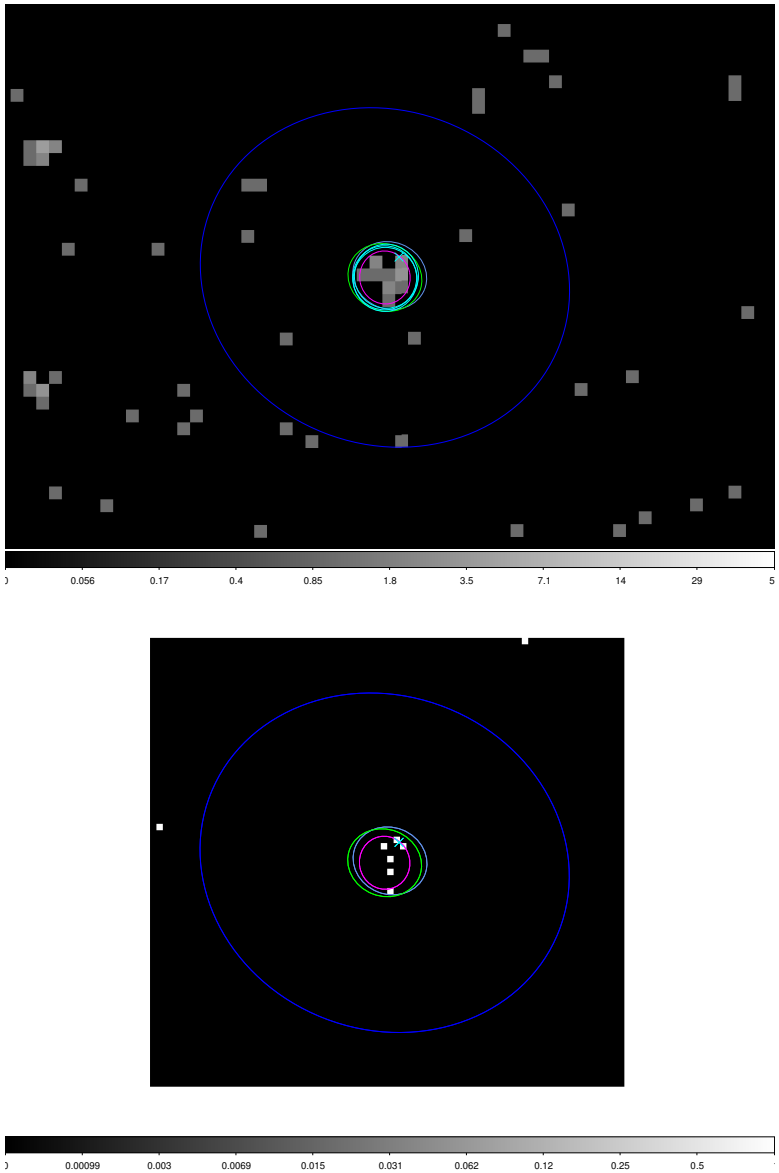

Figure 8: A QA triggered in MRF. The upper image corresponds to the b band image binned at BIN=1, whereas the lower panel corresponds to the m band image, binned at BIN=0.5, which is the resolution at which the fit is performed. Blue ellipses denote the background region, the green ellipse corresponds to the source region, and the magenta ellipse corresponds to the ECF90 region. The green crossed mark the position of the best fit.

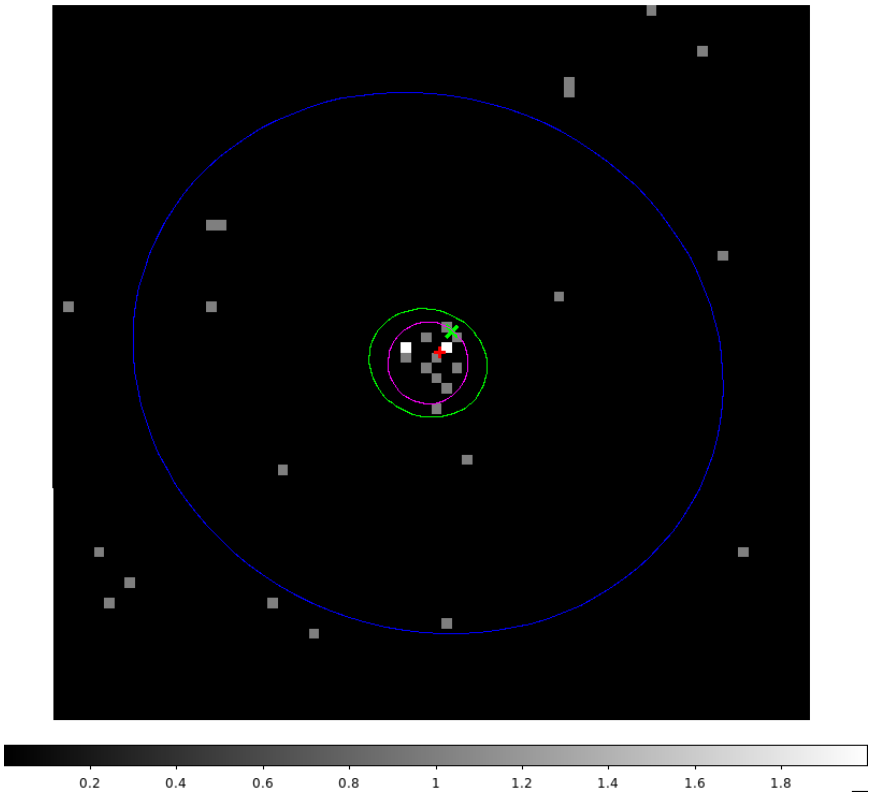

Figure 9: The fit to the same QA case of Figure 8, but this time using the b band and the source region as the notice region. The image has been bined to show the actual resolution at which the fit is performed. The color convention is the same as in the previous figure, whereas the green 'x' and blue cross represent resepctively the original QA fit and the new, improved fit.

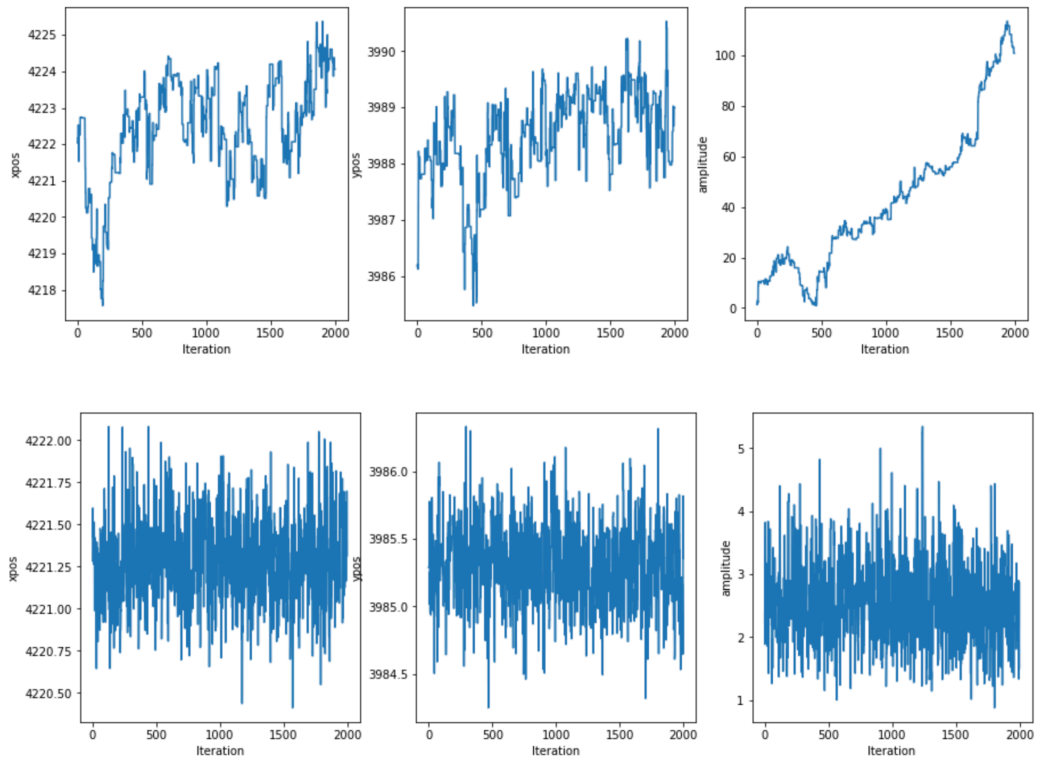

Figure 10: The MCMC traces for the original QA fit (top panel) and the new fit (bottom panel).