Ref: SPIRE-RAL-NOT-003229

Point Source Extraction Design Document

1. Module Information

1.1. Module Owner

Huw Morris (RAL)

1.2. Others Contributing

Rich Savage, Rupert Ward, Anthony Smith (Sussex University)

1.3. Applicable Documents

(List documents here)

1.4. Module Description

The purpose of this module is to extract point sources from a Herschel SimpleImage. Different source extraction algorithms can be used, with the user selecting an algorithm at runtime. The current supported algorithms are DAOPHOT and SussExtractor.

1.5. Input Data Products

image a Herschel SimpleImage.

1.6. Input Parameters

algorithm **Source Extraction Algorithm** The algorithm used to extract point sources. Use "Daophot" or "SussExtractor".

detThreshold **Detector Threshold** The threshold at which a local maximum is determined.

pixelRegion **Pixel Region** The region around each pixel in which to search for local maxima.

fwhm **Full width-half maximum** A Gaussian approximation of the beam profile.
prf **Point response function** A SimpleImage that contains a point response function. If defined, this will be used instead of fwhm.

returnPixelCoordinates optional flag if the found sources are returned in pixel coordinates rather than RA/Dec.

Corner1Ra,
Corner1Dec,
Corner2Ra,
Corner2Dec Restrict the area of search on the image to a rectangle bound by these oppposite corners

inputSourceList **List of input sources** An optional SourceListProduct, containing a list of "known" source positions, at which the fluxes will be extracted.

fluxPriorsLambda **Flux Priors Lambda** Lamba value for the SussExtractor.

fitBackground **Fit Background** Flag for background fit. (SussExtractor only)

subtractMedianBackground **Subtract Median Background** Flag for whether to subtract the median background. (SussExtractor only)

1.7. Output Data Products

result a SourceListProduct. This is a Product which contains an entry for each source found in the image.

2. Module Design

The srcext module is written as a Herschel task.

2.1 Module Method

This module is designed to separate the workings of the task from the actual source extraction as much as possible. The task itself checks input and uses the strategy pattern to create an Extractor, and also creates a SourceExtractorParams.

SourceExtractorParams is a convenience class to store the input parameters of the task.

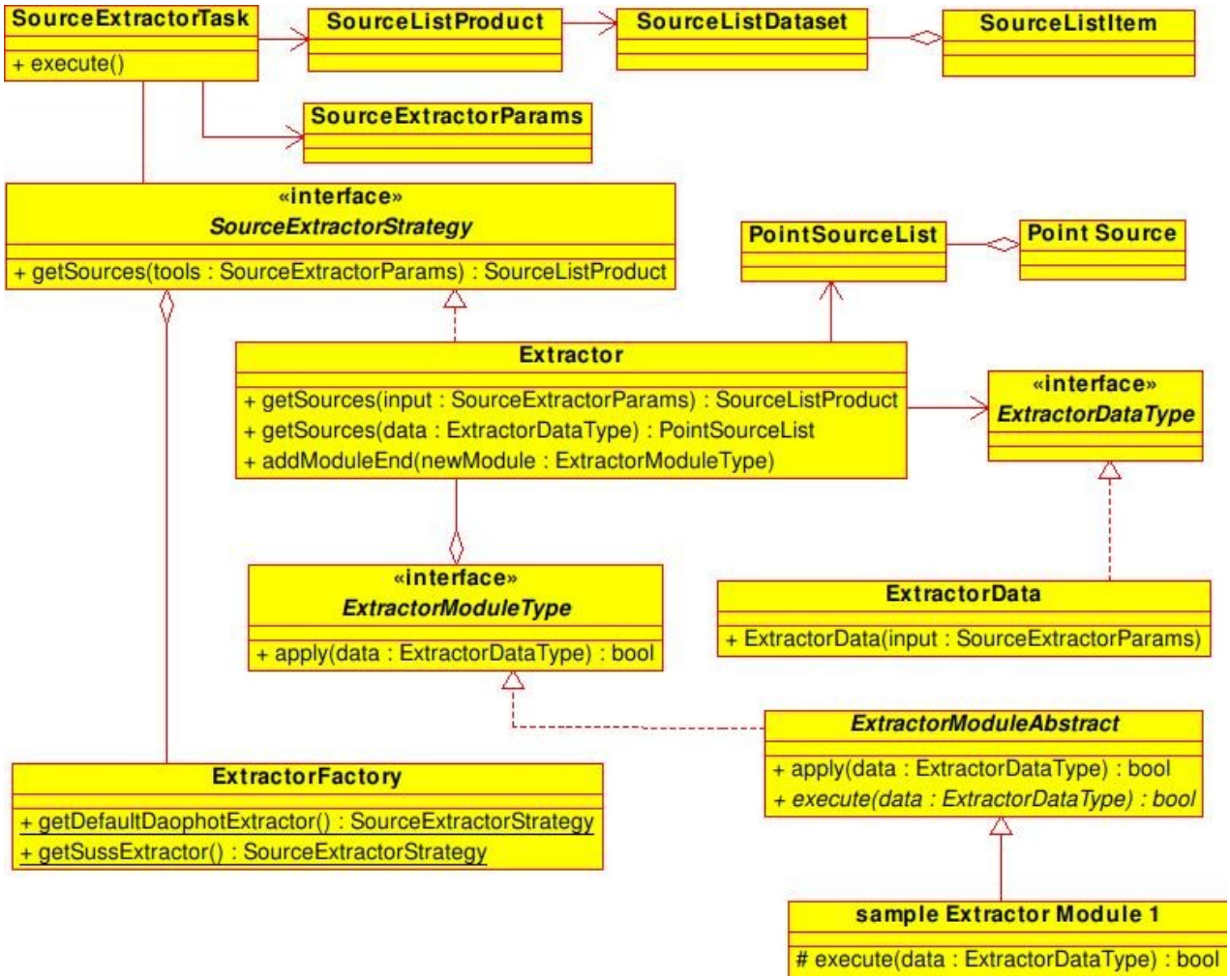
An extractor algorithm is implemented in the Extractor class, which implements

SourceExctractorStrategy. An instance of an extractor contains a number of extractor modules. SourceExtractorFactory uses the factory pattern to return an Extractor with the appropriate modules, depending on the algorithm used.

ExtractorModuleAbstract is the base class for extractor modules, implementing ExtractorModuleType. All extractor modules extend this class. An extractor module performs one step in the overall extraction.

The extractor modules input is ExtractorData, which extends ExtractorDataType.

When the task is executed, an Extractor instance is created, and a sequence of ExtractorModules is added. The SourceExtractorParams is converted into ExtractorData, and each module is executed in turn. If all modules executed without error, a SourceListProduct is created as the output of the task.



Class Diagram of the Source Extraction Module

Note: the existence of PointSourceList, PointSource, and ExtractorData is historical, and doesn't really represent best design. The existing Extractor Modules were written to operate on ExtractorData and a PointSourceList, but new Extractor algorithms only need to implement the SourceExtractorStrategy.

2.2 Improvements and Open Issues

- The list of columns in the SourceListProduct is fixed. Perhaps it would be better if each algorithm could define which columns it needs. In this case, which columns (if any) should be mandatory? Having no mandatory fields would allow for maximum flexibility in the algorithm, but would cause difficulties with any other tool which uses a SourceListProduct as an input.

- Although the architecture allows other algorithms to be added, the original plan was for each extraction algorithm to be a separate class, which would be registered with the task. The user would be able to write a new algorithm (implementing SourceExtractorStrategy), register it with the task, and then immediately use it.

2.3 Problems

The DAOPHOT algorithm appears to be very sensitive to the correct value of fwhm. This may also be true of SussExtractor, but has not yet been properly tested.