Image Segmentation

By David Makovoz

02/04/04

Overview

This document describes the algorithm, I/O of module detect. It was developed to detect point sources or radiation hits in filtered images. The program performs segmentation of an input image read from a standard FITS file. It finds every contiguous cluster of pixels above the user specified threshold and of the user specified size and computes the centroid for each such cluster. It produces a number of products described in this document.

Input

The program reads all the necessary information from a namelist (configuration) file specified on the command line as follows:

```
detect –n namelist.nl
```

Every parameter and file name in the namelist can be also set on the command line. Run detect without any arguments read the tutorial.

Below is an example of a namelist file for module *detect*.

```
&DETECT
  Comment = 'Namelist file for detect',
  Ancillary_File_Path = '/ssc/pipe/davidm/pipe/include',
  Log_Filename = 'stdout',
  FITS_In_Filename = '/ssc/pipe/davidm/sim/2/007.PSP.fits',
  FITS_Out_Filename = '/ssc/pipe/davidm/sim/2/007.detect.fits',
  Mask_Out_Filename = '/ssc/pipe/davidm/sim/2/007.mask.fits',
  Table_Out_Filename = '/ssc/pipe/davidm/sim/2/007.detect.tbl',
  Neighbor_Type =  'sides_and_corners',
  Detection_Max_Area         = 9,
  Detection_Min_Area         = 0,
  Detection_Threshold        = 0.2,
  # Probability_Threshold = 0.5,
  Output_Type             = 2,
  &END
```

Lines beginning with “#” will be ignored. Every line, including the ones beginning with “#” should end with a comma.

File Name	Description
Fits_In_Filename	Name of the fits file with the input image. Required.
Sigma_In_Filename CoverageMap_Filename	The file name of the uncertainty image. Optional The file name of the coverage map. Used optionally for coadded images to adjust the noise level.
N_Masks	Number of mask images, including the P-Mask.
Mask_Filename_#	Filename of a mask image. # runs from 1 to N_Masks. Optional.
Mask_Value_#	A bit-mask value for the pixels in mask image. Required for each Mask_Filename_# given. # runs from 1 to N_Masks.
Fits_Out_Filename	Name of the fits file with the image of the detections. If Output_Type =1 only the centroids of the detected clusters are marked, otherwise all the pixels in the detected clusters are marked. Optional.
Mask_Out_Filename	Name of the fits file with the image of the all the pixels originally detected, even though they might have been later discarded. Optional.
Table_Out_Filename	Output IPAC-table filename. This file contains the list of the cenroids of detected clusters and the value of the greatest pixel in the cluster along with the processing information. Optional.
Peaks_Image_Filename	File name for the output peak pixels image. Applicable for Threshold_Type = "peak" only.
NumberCluster_Out_Filename	File name for the output cluster numbered image.
Complete_Table_Out_Filenam e Bright_Table_Out_Filename	Output IPAC-table filename. This file contains the list of the coordinates and fluxes of all pixels in the detected clusters along with the processing information. Optional. Output IRAC-table filename. This file in addition to the coordinates contains the shape characteristics and total fluxes of the detected objects
Log_Filename	Output log filename. Optional.
Ancillary_File_ Path	Pathname where supporting source files are installed. Optional.

Table 1. Setting file names in the namelist file. The rows with the names or information associated with the input files are shaded.

Parameter Name	Description	Default
Detection_Max_Area	Clusters with number of pixels greater than Detection_Max_Area undergo the iterative process of shrinking/splitting.	9
Detection_Min_Area	Clusters with number of pixels smaller than Detection_Min_Area after the initial thresholding are discarded.	0
Detection_Threshold	Number of *s's* above the mean for Input_Type = "image_input". Directly this number for Input_Type = "snr_input".	0
Input_Type	Determines whether the input file is an image("image_input") or the "signal-to-noise ratio" image corresponding to an image("snr_input")	"image_input"
Output_Type	Determines the output to the fits file given by Fits_Out_Filename. The options are "centroids_only_output", "centroids_and_pixels_output". If the second option is used Complete_Table_Out_Filename can be written.	"centroids_and_pixels_output"
N_Edge	Defines a margin around the edges of the input image; the pixels in the margin are excluded from the computation.	0
Probability_Threshold	A special threshold used for probability images.	0
Neighbor_Type	Determines the way a neighbor pixel is defined for the purposes of identifying a contiguous cluster of pixels. The options are "sides_only" and "sides_and_corners".	"sides_only"
Peaks_Radius	The number of layers of pixels around a pixel to determine whether it is a peak pixel	1
Threshold_Type	Determines the way segmentation threshold is raised. There are three options: "simple", "combo", "peak".	"simple"
Extended_Object_Area	The minimum size of a cluster that won't be split any further, if splitting failed for a particular threshold; applicable for Threshold_Type = "peak" only.	200
Max_Segmentation_Level	Number of iteration over the segmentation threshold; applicable for Threshold_Type = "peak" only.	50

Table 2. Setting parameters in the Namelist file.

Algorithm

The processing stages of module detect are depicted in Figure 2. The program starts by computing the initial threshold based on the parameter `Detection_Threshold` specified by the user. Upon the first pass the program finds all the pixels above the initial threshold. It creates a list of all contiguous clusters of pixels above the initial threshold. Then it compares the number of pixels with the minimum and maximum allowed sizes of a cluster specified by the user through parameters `Detection_Min_Area` and `Detection_Max_Area`. If the number of pixels in a particular cluster is less than the minimum number `Detection_Min_Area` the cluster is discarded. If the number of pixels in a cluster is greater than the maximum number `Detection_Max_Area`, the program goes through an iterative process of raising the threshold with the intention of either shrinking the cluster and/or splitting it into smaller clusters.

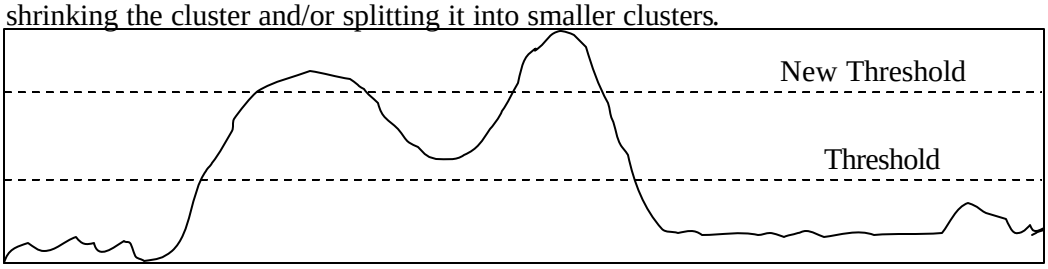

Figure 1 Raising threshold splits a big cluster into two smaller clusters.

To this end the program recalculates the threshold for this particular cluster and finds all the pixels above the new threshold. When the iterative procedure is finished a list of estimated detection locations is created. The centroid is found for each cluster, which is the estimated location of the point source corresponding to this cluster. See Figure 1 for an illustration of this process. Warning, if the number of pixels in a cluster is greater than `MAX_NUM_PIXELS` =10000, which is set in `detect.h` file, the program exits with the appropriate message.

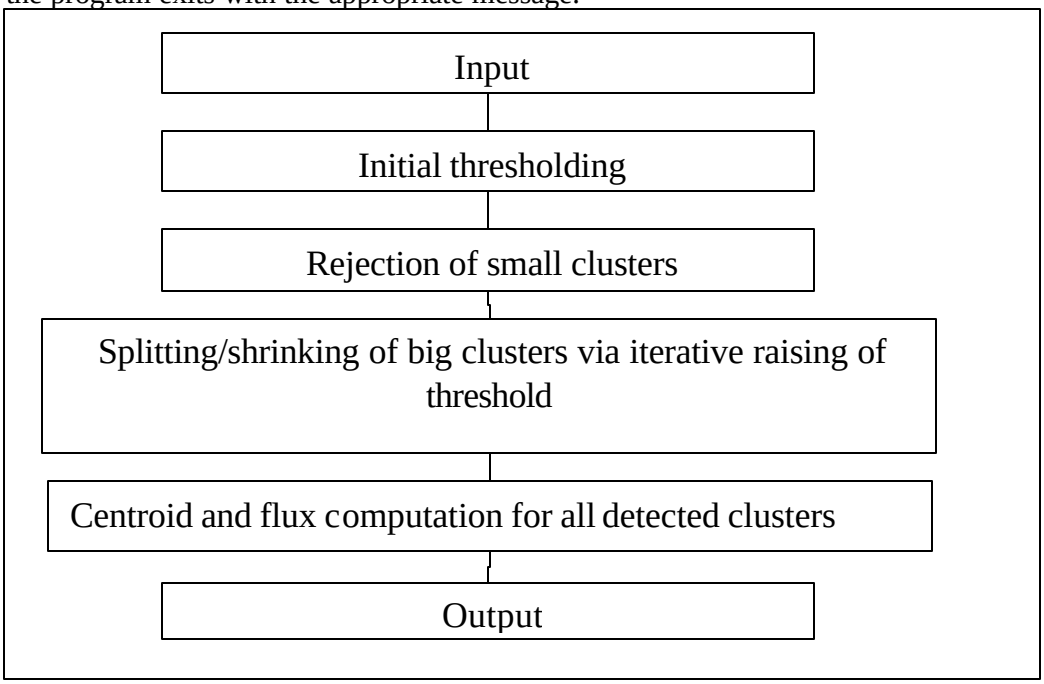

Figure 2 Processing stages of module *detect*.

Cluster

Every pixel in a cluster is greater than the segmentation threshold and is adjacent to at least one pixel in the cluster. The definition of an adjacent pixel depends on the setting of `Neighbor_Type`. It can have two values: "sides_only" (default) and "sides_and_corners". The names are self-explanatory. This setting doesn't affect the way peak pixels are determined (see below); having a common side or corner is always used as the definition of an adjacent pixel for peak pixels.

Thresholding

The calculation of the initial value of the threshold depends on the type of image being processed. It is specified by the user by setting input parameter `Input_Type`. It has two settings: “image_input” (default) and “snr_input”. The first setting corresponds to a regular image, the second one is used for the “signal-to-noise ratio” images. For `Input_Type` = “image_input” the initial threshold T is computed as follows. First the image mean M and standard deviation s are computed. T is set at

$$T = M + D\,s\,,$$

where D is the value of `Detection_Threshold`.
Then M and s are recomputed with the pixel values greater than T excluded from the computation. Then T is recomputed. This is repeated until all the pixels that produce T are below that threshold.
For `Input_Type` = “snr_input” the initial threshold is simply set to this parameter:

$$T = D$$

Initial image segmentation is performed. The clusters exceeding `Detection_Max_Area` size are subject to further segmentation. At this point the threshold is recalculated, so that the new higher threshold will either shrink the "oversized" clusters or break them into several smaller ones. This new threshold is calculated individually for each cluster.
The way the image segmentation threshold is recomputed is determined by the input parameter `Threshold_Type` regardless of what `Input_Type` is set to. There are three settings for `Threshold_Type`: "simple","combo", and "peak".
1. `Threshold_Type` ="simple". The mean value M_{cl} and standard deviation s_{cl} of all pixels in the cluster are calculated. The cluster specific threshold is

$$T_d = M_{cl} + D\,s_{cl}\,,$$

The new threshold is applied to the cluster. If the cluster is shrunk or split, the threshold is recalculated again for each new cluster. If the number of pixels after applying a new threshold doesn’t change, the cluster is passed down for centroid computation, even though the number of pixels in it is greater than `Detection_Max_Area`.
There are two problems with this simple-minded approach. First problem is the fact that once T_d fails to reduce the cluster, this is the end of the segmentation process. The second problem is that this approach will very often fail to resolve two or more point sources that end up in one cluster after the initial thresholding. If one of the point source is significantly brighter than the others it will drive T_d to be higher than the pixels in the fainter point sources. The two schemes below were designed to alleviate these problems.
2. `Threshold_Type` = "combo". The following heuristic formula is used

$$T_{cl} = \sqrt{\left(\left(SegLevel \cdot T_{\min} + Min_{cl}\right) \cdot M_{cl}\right)}$$

Here $T_{\min}$ is the difference between the initial threshold T and the minimum of the whole image. *SegLevel* is the number of times the threshold for this cluster has been raised without any effect on the cluster. Here is how it works. Initially, for each cluster it is set to 1. Then new threshold is calculated. If all the pixels in the cluster are higher than the threshold the threshold is raised and *SegLevel* is incremented. This is repeated until some of the pixels in the cluster end up below the threshold. At this point *SegLevel* is reset to 1. This approach works better then the first one, since the threshold is raised slower and segmentation doesn't stop after the first failure.

3. `Threshold_Type` = "peak" is the most convoluted and is also the most effective one. It is the scheme that has been found to do the best for point source extraction. The condition, that a cluster is split when it is greater than `Detection_Max_Area`, is supplemented with one more condition. A cluster is split as long as there is more then one peak pixel per cluster or it is greater than `Detection_Max_Area`. A peak pixel is any pixel greater than any other pixel within a certain radius. The namelist parameter `Peaks_Radius` is used to specify this radius. By default the radius is 1, i.e. a pixel greater than the 8 adjacent pixels is declared a peak. (Also, if `Peaks_Image_Filename` is set in the namelist an image of peak pixels will be written out). For each cluster the value $P_{\min}$ of the lowest peak is found. Segmentation threshold T_d is first is set to

$$T_{cl} = P_{\min} - T_{\min}$$

If the cluster doesn't change (split or shrink) the threshold is slowly raised:

$$T_{cl} = P_{\min} - \frac{T_{\min}}{SegLevel}$$

This is repeated up to `Max_Segmentation_Level` times. After that the centroid is found for the resulting cluster, even though it might have more then one peak in it. There is one provision to prevent the program from splitting wings off of a bright star. If a cluster fails to be split, normally the threshold will be raised But if the number of pixels in the cluster is greater than `Extended_Object_Area`, then this cluster is left the way it is, even though it might have more than one peak in it.

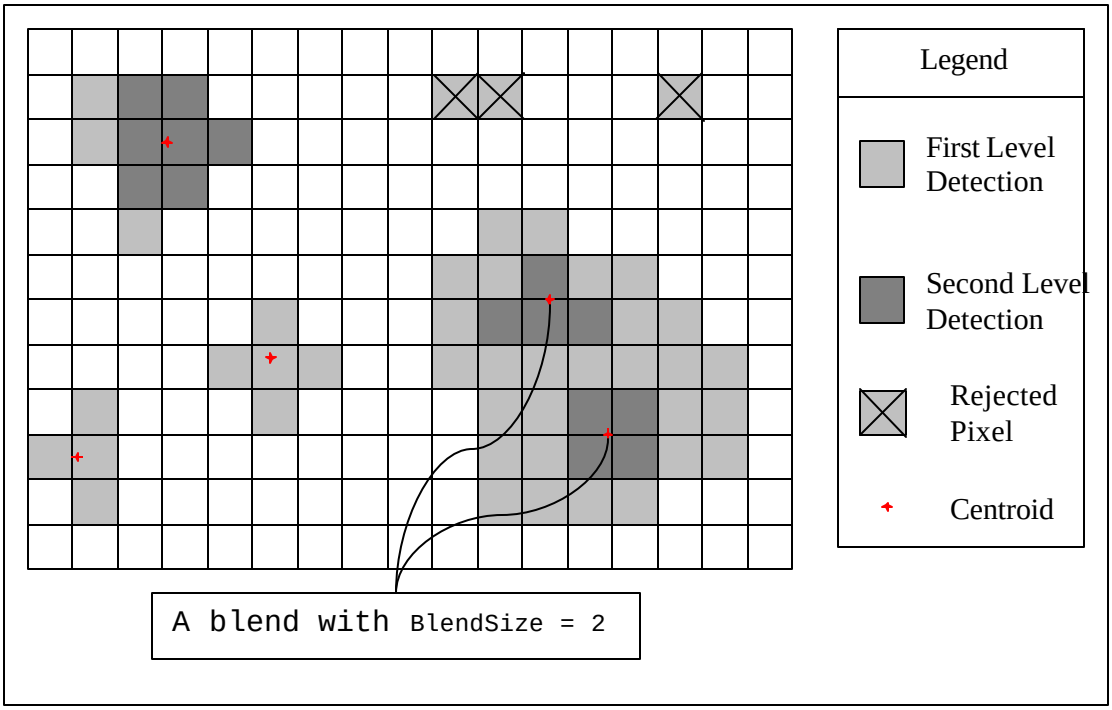

Figure 3 An example of image segmentation with `Detection_Min_Area` = 3; `Detection_Max_Area` = 9. A blend of detections with `BlendSize` = 2 is shown.

Coadded Images

There is one issue with processing coadded images. Due to the variable coverage the noise level, being inversely proportional to the square root of the coverage, varies throughout such an image. One way to deal with this problem is to use *gaussnoise* module to produce an snr image. *gaussnoise* produces a local estimate of the noise and therefore the effects of the variable coverage will be reflected in the snr image. There are two problems with this approach. First, it is time consuming. Second, the process of raising threshold to split/shrink clusters has been designed with the `Input_Type` = "image_type" in mind. It is not clear how it will work for snr images. The alternative is to use a coverage map (`CoverageMap_Filename`). The coverage map is used to attenuate coadded images, i.e. an input coadded image is multiplied by the sqrt(coverage map), if a coverage map is provided.

Probability Images

There is one parameter `Probability_Threshold` that is used for the so-called PSP images produced by the *pointsourceprob* module. The PSP images are products of non-linear filtering of regular images. They have the maximum value of 1. They very often have a cluster of pixels with the values saturated very close to 1. If the probability threshold is set, then pixels greater than the probability threshold are excluded from calculation of the initial threshold. Without using it there is a possibility of having the initial threshold greater then 1, which will lead to having no detections.

Passive deblending

The output of this program is used for point source extraction. Point source extraction performs passive deblending. The detected point sources, determined to be in a close

proximity from one another, so that their PRF's overlap, are fitted simultaneously. This module provides the classification of detections as candidates for passive deblending. If a cluster created by the initial thresholding is consequently split into several clusters, the latter are classified as a blend of clusters. There are two columns in the output table (`Table_Out_Filename`) `Blendid` and `Blendsize` that are used for detection blend classification. `Blendid` keeps track of the sequential number of a detection blend in the table. `Blendsize` gives the number of detections in the blend. The columns have the same values for each detection in a particular blend. For non-blend detections the columns are set to 0. See an example of the output table below.

Centroid

For all detected clusters the centroid is found.

$$Centroid\_X = \frac{\sum_{i \in cluster} X(i) \cdot flux(i)}{\sum_{i \in cluster} flux(i)} \qquad Centroid\_Y = \frac{\sum_{i \in cluster} Y(i) \cdot flux(i)}{\sum_{i \in cluster} flux(i)}$$

The value of the greatest pixel in the cluster is saved as the flux in the output table. If the program is used to detect point sources, the above quantity can be used only as a guide to what the flux of the point source should be, since the program is not meant to compute the photometry on the point sources. The modules *sourcesimate* or *aperture* should be used for this purpose.

Computing Shape Parameters

If `Bright_Table_Out_Filename` is requested, the following parameters are computed for each cluster in addition of it's centroid: total *flux*, semi-major axis length *maj_ax*, semi-minor length *min_ax*, the angle between the major axis and the `NAXIS1` image axis *theta*, *elongation*, and *ellipticity*.

$$X^2 = \frac{\sum_{i \in cluster} \left(X(i) - Centroid\_X\right)^2 \cdot flux(i)}{\sum_{i \in cluster} flux(i)}; \qquad Y^2 = \frac{\sum_{i \in cluster} \left(Y(i) - Centroid\_Y\right)^2 \cdot flux(i)}{\sum_{i \in cluster} flux(i)};$$

$$XY = \frac{\sum_{i \in cluster} \left(X(i) - Centroid\_X\right) \cdot \left(Y(i) - Centroid\_Y\right) flux(i)}{\sum_{i \in cluster} flux(i)};$$

$$maj\_ax^2 = \frac{X^2 + Y^2}{2} + \sqrt{\left(\frac{X^2 - Y^2}{2}\right)^2 + (XY)^2};$$

$$\min\_ax^2 = \frac{X^2 + Y^2}{2} - \sqrt{\left(\frac{X^2 - Y^2}{2}\right)^2 + (XY)^2};$$

$$theta = \frac{1}{2}\arctan\frac{2 \cdot XY}{X^2 - Y^2};$$

$$elongation = \frac{maj\_ax}{\min\_ax}; \qquad ellipticity = 1 - \frac{\min\_ax}{maj\_ax}.$$

If the uncertainty image is provided then the standard deviation is computed for the centroid, flux, semi-major and semi-minor axes length, and theta: *delta_x, delta_y, delta_xy, delta_flux, delta_maj_ax, delta_min_ax, delta_theta*.

$$delta\_x = \sqrt{\frac{\sum_{i \in cluster} (X(i) - Centroid\_X)^2 \cdot sigma(i)}{\left(\sum_{i \in cluster} flux(i)\right)^2}};$$

$$delta\_y = \sqrt{\frac{\sum_{i \in cluster} (Y(i) - Centroid\_Y)^2 \cdot sigma(i)}{\left(\sum_{i \in cluster} flux(i)\right)^2}};$$

$$\mathrm{cov}\_xy = \frac{\sum_{i \in cluster} (X(i) - Centroid\_X)(Y(i) - Centroid\_Y) \cdot sigma(i)}{\left(\sum_{i \in cluster} flux(i)\right)^2};$$

$$delta\_xy = sign(\mathrm{cov}\_xy)\sqrt{|\mathrm{cov}\_xy|};$$

$$delta\_maj\_ax^2 = \frac{(delta\_x)^2 + (delta\_y)^2}{2} + \sqrt{\left(\frac{(delta\_x)^2 - (delta\_y)^2}{2}\right)^2 + (\mathrm{cov}\_xy)^2};$$

$$delta\_\min\_ax^2 = \frac{(delta\_x)^2 + (delta\_y)^2}{2} - \sqrt{\left(\frac{(delta\_x)^2 - (delta\_y)^2}{2}\right)^2 + (\mathrm{cov}\_xy)^2};$$

$$delta\_theta = \frac{1}{2}\arctan\frac{2 \cdot \mathrm{cov}\_xy}{(delta\_x)^2 - (delta\_y)^2};$$

The definitions above are identical to those used by SExtractor by E.Bertin.
In additions to the above the following parameters are computed and written in the table output: *n_pixels*, *perimeter*, and *linearity*.
Parameter *n_pixels* is simply the number of pixels in the cluster. Parameter *perimeter* is defined as the number of sides of pixels that border on the pixels which are not part of the cluster.
Parameter *linearity* is meant to indicate how close to a line the cluster is:

$$linearity = \frac{perimeter}{4\sqrt{n\_pixels}}$$

It is defined in such a way, that it is equal to 1 for square clusters and it is ~ *0.5*n_pixels* for perfectly straight line clusters. Parameter *perimeter* will also be big for "edgy" objects like bright point sources with the diffraction pattern.
To illustrate the above definitions here is a table with these parameters for the clusters in Figure 7:

Cluster number	*n_pixels*	*perimeter*	*linearity*
1	7	12	1.13
2	5	12	1.34
3	4	9 (the edge side excluded)	1.125
4	4	10	1.25
5	4	8	1

Output

Module *detect* generates the following output:

1. `Table_Out_Filename`. An IPAC table with the centroids and greatest fluxes for all clusters. Below is a sample of the table output:

```
\char comment = Output from DETECT, version  1.00
\char Date-Time = Mon Dec  4 18:12:51 2000
\char FITS_In_Filename = 001.stel.xgal.xtnd.noise.fits
\char Table_Out_Filename = 001.stel.xgal.xtnd.noise.tbl
\int Detection_Max_Area = 9
\int Detection_Min_Area = 0
\float Detection_Threshold = 0.825842
\int Input_Type = 2
\int Number_Of_Detections = 275
|     srcid|         x|         y|      flux|   BlendId| BlendSize|
|  i       |  r       |  r       |  r       |         i|         i|
       0     2.500e+00  2.295e+02  1.328e+03          0          0
       1     3.500e+00  7.750e+01  1.369e+03          0          0
       2     3.500e+00  1.905e+02  1.338e+03          1          2
       3     4.825e+00  1.902e+02  1.435e+03          1          2
       4     4.500e+00  1.115e+02  1.332e+03          0          0
       5     8.403e+00  2.050e+02  1.275e+04          2          3
       6     5.500e+00  2.050e+02  1.330e+03          2          3
       7     6.385e+00  2.067e+02  1.853e+03          2          3
       8     5.500e+00  1.275e+02  1.328e+03          0          0
…
```

2. `Complete_Table_Out_Filename`. Optionally, if `Output_Type` =2, an IPAC table with the coordinates of all detected pixels and their values. Below is a sample of the complete table output:

```
\char comment = Output from DETECT, version  1.00
\char Date-Time = Wed Dec 13 10:26:50 2000
\char FITS_In_Filename = /ssc/pipe/davidm/sim/007.PSP.fits
\char Mask_Out_Filename = /ssc/pipe/davidm/sim/007.mask.fits
\char FITS_Out_Filename = /ssc/pipe/davidm/sim/007.detect.fits
\char Table_Out_Filename = /ssc/pipe/davidm/sim/007.detect.tbl
\char Complete_Table_Out_Filename =
/ssc/pipe/davidm/sim/007.complete_detect.tbl
\char Mask_Out_Filename = /ssc/pipe/davidm/sim/007.mask.fits
\int Detection_Max_Area = 9
\int Detection_Min_Area = 0
\float Detection_Threshold = 0.2
\int Input_Type = 2
\int Output_Type = 2
\int Number_Of_Detections = 237
|     srcid|         x|         y|      flux|
|  i       |  i       |  i       |  r       |
       0          1        162     1.339e-01
       0          0        162     1.671e-01
       0          0        163     1.547e-01
       0          1        163     1.289e-01
       1          3        106     1.115e-01
       2          8        119     3.098e-01
       2          9        118     7.262e-01
```

3. `Fits_Out_Filename`. A FITS image of the detected pixel clusters. If `Output_Type` = 1 the pixels corresponding to the centroids are set to 1, all other

pixels are set to 0.

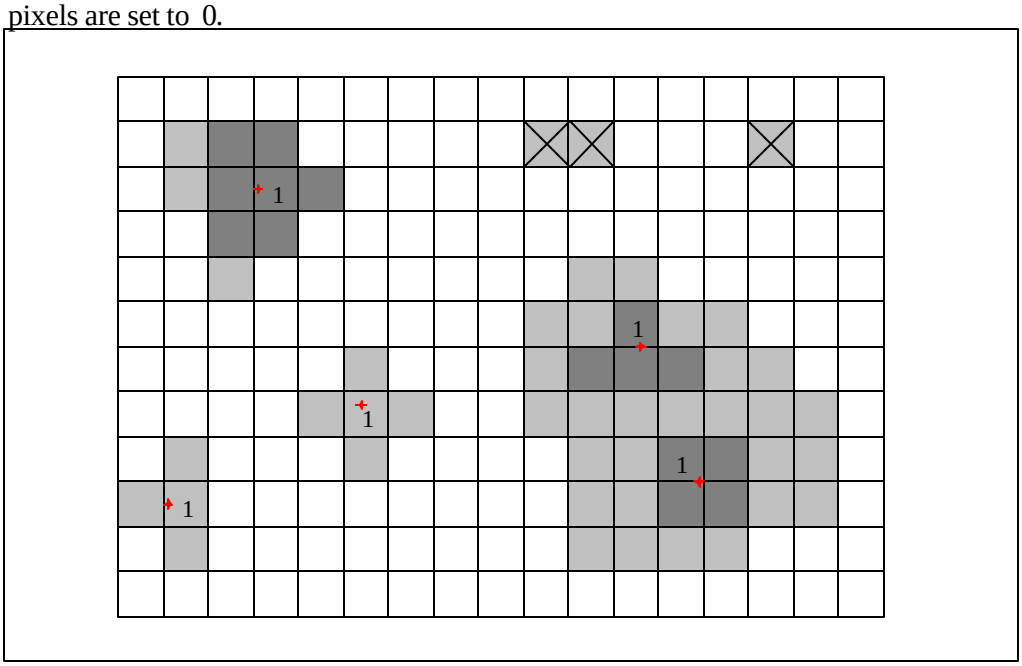

Figure 4. Fits Output for `Output_Type` = 1 corresponding to the segmented image in Figure 3. The pixels without numbers are set to 0.

If `Output_Type` = 2 all the pixels in the detected clusters are set to their segmentation level. The difference between this output and the Mask File (see next) is that this image has only the pixels detected above the final threshold, whereas in the Mask File all the pixels detected at least once are marked regardless of their final status.

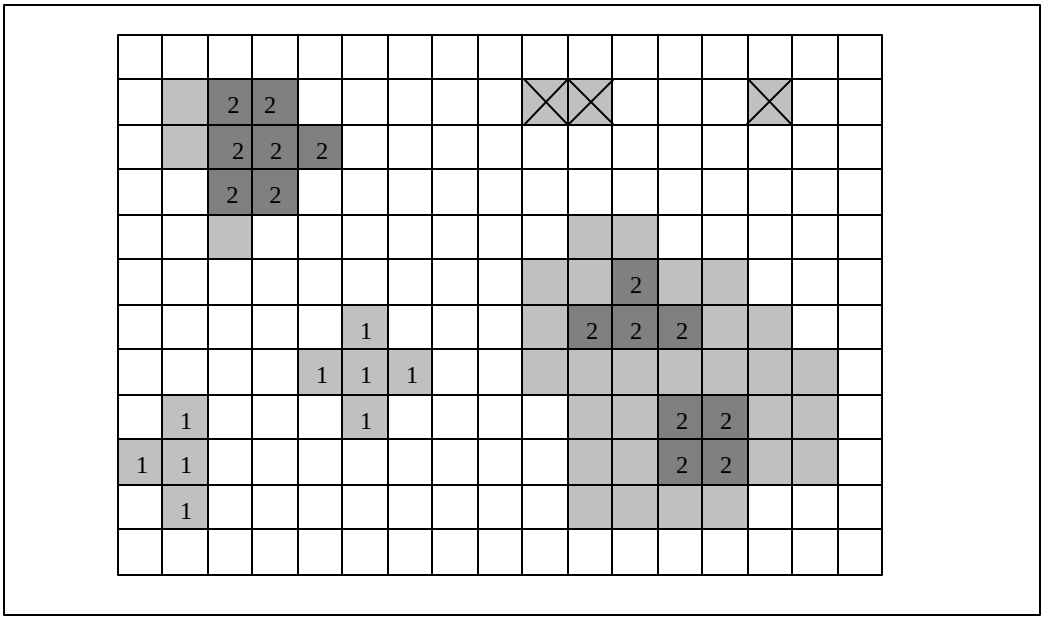

Figure 5. Fits Output for `Output_Type` = 2 corresponding to the segmented image in Figure 3

4. `Mask_Out_Filename`. A FITS image that marks each pixel detected during the initial thresholding. Each pixel is set to the highest segmentation level for which it has been detected above the corresponding threshold.

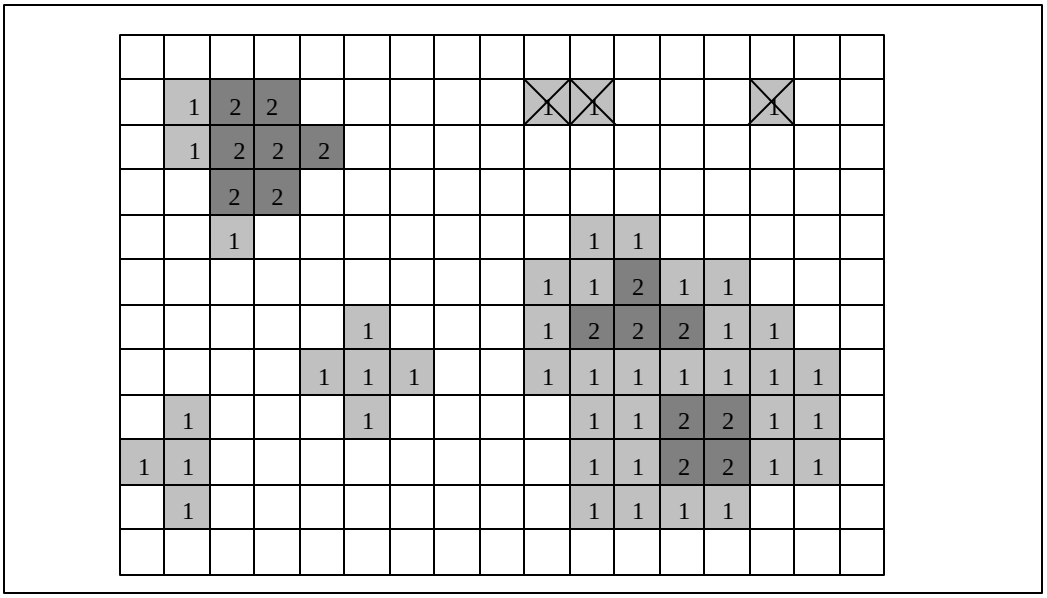

Figure 6. Mask FITS output corresponding to the segmented image in Figure 3.

5. `Peaks_Image_Filename`. A FITS image of peak pixels. See above the definition of a peak pixel. Each peak pixel is set to 1. All other pixels are set to 0.

6. `NumberCluster_Out_Filename`. A FITS image of the detected pixels. Unlike `Fits_Out_Filename`, each pixel in a cluster is set to the sequential number of the cluster in the list of all clusters. This image is used as an intermediate product in the Dual Outlier Detection scheme.

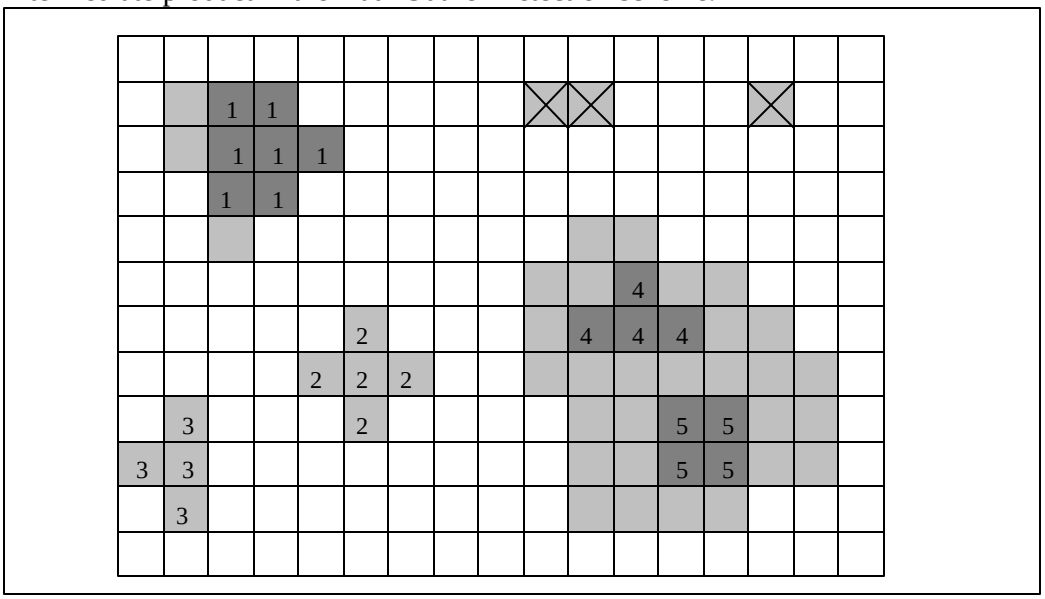

Figure 7. NumberCluster output corresponding to the segmented image in Figure 3.

7. A log file containing processing statistics and status messages.

8. `Bright_Table_Out_Filename`. This table contains the shape characteristics along with the centroid, total flux, and the total number of pixels per detectioin. Note that the column name for ellipticity is `ellipticit` because of the column width restriction. The units of *theta* and *delta_theta* are degrees, the flux is in the units of the input image, the coordinates are in pixels.

```
\char comment = Output from DETECT, version  4.00
\char Date-Time = Wed Feb  4 12:12:34 2004
\char comment = dlimage library version  2.00
\char FITS_In_Filename = apex_small/Coadd/coadd_Tile_001_Image_minback.fits
\char FITS_Out_Filename = apex_small/Coadd/coadd_Tile_001_Image_detect_bright_mask.fits
\char Sigma_In_Filename = apex_small/Coadd/coadd_Tile_001_Unc.fits
\char Bright_Table_Out_Filename = apex_small/Coadd/coadd_Tile_001_Image_detect_bright.tbl
\float Tile_OffsetX = 0
\float Tile_OffsetY = 0
\int Peaks_Radius = 1
\char CoverageMap_Filename = apex_small/Coadd/coadd_Tile_001_Cov.fits
\float Min_Coverage = 2
\float Effective_Threshold = 0.259142
\int Extended_Object_Area = 200
\int Max_Segmentation_Level = 50
\int Detection_Max_Area = 1000
\int Detection_Min_Area = 30
\int N_Edge = 0
\float Detection_Threshold = 6
\char Input_Type = image_input
\char Output_Type = centroids_and_pixels_output
\char Neighbor_Type = sides_only
\char Threshold_Type = peak
\int Number_Of_Detections = 118
| srcid|          x|   delta_x|          y|   delta_y|   delta_xy|         flux|  delta_flux|       maj_ax|   delta_maj|       min_ax|   delta_min|  elongation| ellipticity|         theta| delta_theta| n_pixels|   perimeter|   linearity|
|i    |r          |r         |r          |r         |r          |r            |r           |r             |r           |r             |r           |r           |i           |r             |r           |i        |r           |            |
       1  6.292e+02  6.751e-02  2.552e+01  7.353e-02  3.333e-02  8.223e+02  1.407e+01  3.118e+00  7.856e-02  2.515e+00  6.159e-02  1.240e+00  1.934e-01  5.636e+01  5.636e+01       332        94  1.290e+00
       2  1.493e+02  1.350e-02  3.395e+01  1.430e-02 -4.271e-03  3.814e+03  2.266e+01  1.553e+00  1.465e-02  1.471e+00  1.312e-02  1.056e+00  5.289e-02 -6.515e+01 -6.063e+01       278       132  1.979e+00
       3  1.232e+03  3.253e-02  3.493e+01  3.006e-02  1.101e-02  3.814e+02  7.638e+00  1.118e+00  3.354e-02  1.020e+00  2.893e-02  1.095e+00  8.712e-02  1.720e+01  4.057e+01        70        46  1.375e+00
```