CCLRC / Rutherford Appleton Laboratory
Particle Physics & Astronomy Research Council
Starlink Project
Starlink User Note 255.1

SUN/255.1

D.S. Berry
6th March 2009

CUPID

Version 0.0-38

Users' Manual

Abstract

CUPID is a package of the identification and analysis of clumps of emission within 1-, 2- and 3-dimensional data arrays.

Contents

1 Introduction

The CUPID package provides a set of tools for identifying and analysing clumps of emission within 1, 2 or 3D data arrays. Specifically, it currently provides the following facilities:

- background estimation (uses spatial filtering to remove features with scale size less than a given value).
- identification of clumps of emission using a variety of different algorithms (*e.g.* Gauss-Clumps, ClumpFind, *etc*).
- creation of catalogues of clump parameters
- creation of cut-out images containing the emission from individual clumps.

CUPID processes data in *NDF* format. This is the standard data format used by most Starlink software, and is described fully in SUN/33. However, other astronomical data formats may also be processed using transparent on-the-fly data conversion facilities provided by the NDF subroutine library, and the CONVERT package. The use of these facilities is described in section **??**, and more fully in SUN/55.

All CUPID applications use standard Starlink subroutine libraries for accessing parameters, producing graphics, reporting errors, *etc*. They therefore look and feel very similar to applications in other Starlink packages such as KAPPA and CCDPACK. The following sections in SUN/95 (the KAPPA manual) should therefore be consulted for general information about these issues:

- "Bad pixel values"
- "Parameters"
- "Graphics Devices and Files"
- "Plotting Styles and Attributes"
- "Data Structures"
- "NDF Sections"
- "NDF History"
- "The Graphics Database"
- "Using World Co-ordinate Systems"
- "Procedures"

1.1 Available Commands

Currently, the following commands are available within CUPID:

CUPIDHELP - gives on-line help for all CUPID commands;

EXTRACTCLUMPS - calculates parameters for the clumps within a 1, 2 or 3D data array, given a mask array that identifies the pixels within each clump;

FINDBACK - applies spatial filtering to a 1, 2 or 3D array in order to remove features smaller than a specified size and thus produce an estimate of the background within the NDF;

FINDCLUMPS - finds clumps within a 1, 2 or 3D array and produces a mask identifying the pixels included in each clump. Also produces a catalogue of clump parameters. Various clump identification algorithms are supported such as ClumpFind, GaussClumps, *etc.*

MAKECLUMPS - creates a 1, 2 or 3D array containing a collection of random Gaussian clumps, with added noise.

OUTLINECLUMP - draw an outline on a previously displayed image around a 2-dimensional clump.

2 Clump Identification Algorithms

This section lists and describes the clump identification algorithms which are implemented within the CUPID FINDCLUMPS command.

Note, it is assumed that any background extended emission has been removed from the data before using FINDCLUMPS. This can be done using the FINDBACK command.

2.1 GaussClumps

This is based on the algorithm described by *Stutzki & Gusten* (1990, ApJ 356, 513). This algorithm proceeds by fitting a Gaussian profile to the brightest peak in the data. It then subtracts the fit from the data and iterates, fitting a new ellipse to the brightest peak in the residuals. This continues until any one of the "termination criteria" described below is satisfied. Each fitted ellipse is taken to be a single clump and is added to the output catalogue. In this algorithm, clumps may overlap, and for this reason each input pixel cannot simply be assigned to a single clump (as can be done for algorithms such as FellWalker or ClumpFind). Therefore, when using GassClumps, the primary output NDF from FINDCLUMPS does not hold the clump index at each input pixel. Instead it holds the sum of all the fitted Gaussian that contribute to each input pixel position.

Any input variance component is used to scale the weight associated with each pixel value when performing the Gaussian fit. The most significant configuration parameters for this algorithm are: `GaussClumps.FwhmBeam` and `GaussClumps.VeloRes` which determine the minimum clump size, and `GaussClumps.Thresh` which (together with the ADAM parameter RMS) determine the third of the termination criteria listed below.

Note, this implementation of the GaussClumps algorithm is a completely independent re-write, and includes some differences from other GaussClumps implementations. Specifically, these include the following modifications.

- The Gaussian fitting is based on the SUMSL module (algorithm 611) from the TOMS library available from http://www.netlib.org/.
- Any available variance information is used to weight the pixels when doing the Gaussian fit. This is in addition to the Gaussian weighting function implied by configuration parameters `GaussClumps.Wwidth` and `GaussClumps.Wmin`.
- The termination criteria are different. FINDCLUMPS stops finding further clumps if any one of the following criteria is met.

  1. the total data sum in the fitted Gaussians is equal to or exceeds the total data sum in the supplied input data (this is the original termination criterion used by *Stutzki & Gusten*).
  2. The number of clumps already found equals the value of configuration parameter `GaussClumps.MaxClumps`.
  3. The number of consecutive fitted peaks with peak value below the value of configuration parameter `GaussClumps.Thresh` reaches the value of configuration parameter `GaussClumps.NPad` (the final group of NPad clumps are not included in the returned list of usable clumps).
  4. The number of failed attempts to fit consecutive clumps reaches the value of configuration parameter `GaussClumps.MaxSkip`.

- A clump will be ignored if its fitted peak value is a long way above or below the peak value of the previously fitted clump. The definition of "a long way" is more than `GaussClumps.NSigma` times the standard deviation of the previous `GaussClumps.NPeak` fitted peaks. This restriction is only imposed once `GaussClumps.NPeak` peaks have been fitted.
- In certain situations the chi-squared value that is minimised when fitting a Gaussian clump to a peak in the data array may be dominated by pixels that are largely unaffected by changes in the parameters of the Gaussian clumps[1]. This can result in a very poor fit to the clump. To avoid this, an attempt is made to identify such pixels and to lower the weight associated with them.

2.2 ClumpFind

This algorithm was developed by Jonathan Williams and had been described fully in *Williams, de Geus & Blitz* (1994, ApJ 428, 693).

Briefly, it contours the data array at many different levels, starting at a value close to the peak value in the array and working down to a specified minimum contour level. At each contour level, all contiguous areas of pixels that are above the contour level are found and considered in turn. If such a set of pixels includes no pixels that have already been assigned to a clump

[1]This can happen for instance when neighbouring peaks are present within the area of pixels used to fit a central peak.

(*i.e.* have already been identified at a higher contour level), then the set is marked as a new clump. If the set includes some pixels that have already been assigned to a clump, then, if all such pixels belong to the same clump, that clump is extended to include all the pixels in the set. If the set includes pixels that have already been assigned to two or more clumps, then the new pixels in the set are shared out between the two or more existing clumps. This sharing is done by assigning each new pixel to the closest clump. Note, this is based on the distance to the nearest pixel already assigned to the clump, not the distance to the central or peak pixel in the clump. The above paper refers to this as a "friends-of-friends" algorithm.

This process continues down to the lowest specified contour level, except that new clumps found at the lowest contour level are ignored. However, clumps found at higher contour levels are followed down to the lowest specified contour level.

2.2.1 Comparing CUPID ClumpFind with other Implementations

The CUPID implementation of ClumpFind is a completely independent and total re-write, based on the description of the algorithm in the 1994 Williams, de Geus & Blitz paper. Consequently, it differs slightly from other implementations such as the IDL implementation available from Williams own web page at http://www.ifa.hawaii.edu/ jpw/). In particularly, several extra configuration parameters have been added by CUPID to provide more detailed control of the algorithm. If you want to do a direct comparison between the CUPID implementation of the ClumpFind algorithm and another implementation, then you should set these extra parameters to the following values:

- `ClumpFind.FwhmBeam` = 0
- `ClumpFind.MaxBad` = 1.0
- `ClumpFind.MinPix` = 6 (for 3D data - use 20 for 2D data)
- `ClumpFind.VeloRes` = 0
- `ClumpFind.IDLAlg` = 1
- `ClumpFind.AllowEdge` = 1

You should also note that the pixel co-ordinate system used by the two implementations differ, and consequently the positions reported for the clump peaks will also differ. The IDL implementation of ClumpFind uses a pixel coordinate system in which the first pixel (*i.e* the bottom left pixel of a 2D image displayed "normally") is centred at (0,0). This differs from both the FITS and NDF definition of pixel co-ordinates. In FITS, the centre of the first pixel in the array has a value of 1.0 on every pixel axis. In NDF, the centre of the first pixel has a value of $LBND(I) - 0.5$ on axis I, where $LBND(I)$ is an attribute of the NDF known as the "pixel origin". For an NDF which has been derived from a FITS file and which has been subjected to no geometric transformations, the pixel origin will be 1.0 on every pixel axis, resulting in the centre of the first pixel having a value of 0.5 on every pixel axis.

Some implementations of ClumpFind do not conform exactly to the description in the published paper. Specifically:

1. the way in which areas containing merged clumps are divided up between individual clumps can differ slightly

2. the restriction that all peaks must extend at least as far as the second contour level is sometimes omitted

3. the restriction on the minimum number of pixels contained within a clump is sometimes varied in value

CUPID provides an option to use either the published algorithm, or the algorithm implemented by the IDL package available from Jonathan Williams WWW site (as available at 28th April 2006). Setting the configuration parameter `ClumpFind.IDLAlg` to a non-zero value will cause the IDL implementation to be used. The default value of zero causes the published algorithm to be used.

2.3 Reinhold

This algorithm was developed by Kim Reinhold whilst working at the Joint Astronomy Centre in Hilo, Hawaii. Its overall strategy is first to identify pixels within the data array which mark the edges of clumps of emission. This typically produces a set of rings (in 2D), or shells (in 3D), outlining the clumps. However, these structures can be badly affected by noise in the data and so need to be cleaned up. This is done using a sequence of cellular automata which first dilate the rings or shells, and then erodes them. After cleaning, all pixels within each ring or shell are assumed to belong to a single clump.

2.3.1 Identifying the Clump Edges

The initial identification of the edges is done by considering a set of 1-dimensional profiles through the data array. Each such profile can be represented by a plot of data value against distance (in pixels) along the profile. For each such profile, the algorithm proceeds as follows.

1. Find the highest remaining (*i.e.* unused) data value in the profile.

2. If this value is less than a specified background level (given by the configuration parameter `Reinhold.Noise`), then there are no remaining significant peaks in the profile so continue with the next profile.

3. Work out away from the peak position along the profile in both directions to find the edges of the peak. A peak ends when it either i) meets a pixel which has already been included within another peak, or ii) two adjacent pixels are both below the background level, or iii) the average gradient of the profile over three adjacent pixel drops below a minimum value specified by the configuration parameter `Reinhold.FlatSlope`, or iv) the end of the profile is reached.

4. If the peak was not truncated by reaching either end of the profile, and if the peak spans sufficient pixels, the positions of the two edges of the peak are marked in a mask array which is the same shape and size as the 2D or 3D data array. The minimum number of pixels spanned by a peak in order for the peak to be usable is given by the configuration parameter `Reinhold.MinPix`.

5. The position of the peak itself is also marked so long as its peak value is above a specified minimum value (given by configuration parameter `Reinhold.Thresh`).

6. The pixels within the 1D profile which fall between the upper and lower edges of the peak are marked as "used", and the algorithm loops back to the start (*i.e.* step 1 above).

This algorithm is applied to each 1D profile in turn. These profiles are divided into groups of parallel profiles; the first group contains profiles which are parallel to the first pixel axis, the second group contains profiles which are parallel to the second pixel axis, *etc.* There are also groups which contain parallel profiles which proceed diagonally through the data array. Thus there is a group of parallel profiles for every pixel axis and for every possible diagonal direction. Within each group, there are sufficient profiles to ensure that every element in the data array is included in a profile within the group.

Once all profiles have been processed, a 2 or 3D array is available that identifies both the edges of the peaks and also the peak positions themselves. Pixels which are flagged as peaks are only retained if the pixel was found to be a peak in every profile group. That is, pixels which appear as a peak when profiled in one direction but not when profiled in another are discarded.

2.3.2 Cleaning the Clump Edges

The initial identification of clumps edges results in a mask array in which each data pixel is marked as either an edge pixel or a peak pixel or neither. Usually, the edge pixels can be seen to follow the outline of the visible clumps, but will often be badly affected by noise in the data. For instance, there may be holes in the edges surrounding a peak, or spurious pixels may have been marked as edge pixels. Before continuing, it is necessary to reduce the effect of this noise. This is done in two steps described below.

1. The edge regions are "dilated" (*i.e.* thickened) using a cellular automata algorithm which proceeds as follows: if a pixel is marked as an edge pixel, then all immediate neighbours of the pixel are also marked as edge pixels. Each pixel is considered to be the central pixel in a square of 3×3 neighbouring pixels for 2D data, or the central pixel in a cube of 3×3×3 neighbouring pixels for 3D data.

2. The thickened edge regions are then "eroded" (*i.e.* made thinner) using another cellular automata algorithm which proceeds as follows. If the number of neighbouring edge pixels surrounding a central pixel is greater than a specified threshold value (given by the configuration parameter `Reinhold.CAThresh`), the central pixel is marked as an edge pixel. If the number of neighbouring edge pixels is equal to or below this threshold, the central pixel is not marked as an edge pixel. This transformation can be applied repeatedly to increase the amount of erosion by setting a value greater than one for the configuration parameter `Reinhold.CAIterations`.

2.3.3 Filling the Clump Edges

Once the edges have been cleaned up, the volume contained within the edges can be filled with an integer which identifies the associated peak. This algorithm proceeds as follows.

1. The mask array is scanned for pixels which are marked as peaks. Recall that only those pixels which are seen to be peaks when profiled in all directions have been retained. Each of these pixels thus represents a local maximum in the data value, and has a significantly higher data value than any of the surrounding pixels. Each such peak is given a unique integer identifier. This identifier is used within the following steps to label all pixels in the clump of emission surrounding the peak.

2. A line of pixels parallel to the first pixel axis, and which passes through the peak, is then considered. The line is followed away from the peak, in both directions, until pixels are encountered which are flagged as edge pixels. All the pixels along this line between the two edge pixels are assigned the clump identifier associated with the central peak.

3. For each such pixel, another line of pixels parallel to the second pixel axis and passing through the pixel is considered. The line is followed away from the pixel, in both directions, until edge pixels are encountered. All the pixels along this line between the two edge pixels are also assigned the clump identifier associated with the central peak.

4. For each such pixel, another line of pixels parallel to the third pixel axis and passing through the pixel is considered. The line is followed away from the pixel, in both directions, until edge pixels are encountered. All the pixels along this line between the two edge pixels are also assigned the clump identifier associated with the central peak.

The above process will fill the volume between the edges, but may miss some pixels (*e.g.* if the initial line parallel to the first pixel axis spans the clump at an unusually narrow point). In order to alleviate this effect, the above process is repeated, but scanning the pixels axes in a different order (2,3,1 instead of 1,2,3). For 3D data, it is repeated a third time using the axis order (3,1,2).

Even so, it should be recognised that this filling algorithm will fail to fill certain shapes entirely. For instance, "S"-shaped clumps could not be filled completely using this algorithm.

2.3.4 Cleaning up the Filled Clumps

The use of cellular automata to clean up the edges reduces the likelihood of "holes" within the clump edges, but does not eliminate this risk entirely. When the clump-filling algorithm described above encounters a hole in the edges surrounding a peak, the clump identifier value will "leak out" through the hole into the surrounding areas. This is where the limitations of the filling algorithm have a positive benefit, in that they prevent the leak from spreading round corners without limit. Instead, such leaks will tend to produce straight features radiating out from a clump parallel to a pixel axis, which will terminate as soon as they meet another edge.

It is thus possible for the two or more clumps to "claim" a given pixel. This will happen if there are holes in the edges surrounding the peaks which allow the filling process to leak out. In this case, each pixel is assigned to the clump associated with the nearest peak.

Another cellular automata is used once the filling process has been completed to reduce the artifacts created by these leaks. This cellular automata replaces each clump identifier by the most commonly occurring clump identifier within a 3×3×3 cube (or 3×3 square for 2D data) of neighbours. This process can be repeated to increase the degree of cleaning, by assigning a value greater than one to the configuration parameter `Reinhold.FixClumpsIterations`.

The results of this cleaning process are the final clump allocations for every data pixel, from which the catalogue of clump parameters is produced.

2.4 FellWalker

This algorithm is most simply described assuming the data array is 2-dimensional, although it applies in a similar manner to 3-dimensional data. Its name was chosen to suggest a parallel between a contour map of a 2D astronomical data array, and the height contours seen in a geographical map of a hilly area, such as used by most fell-walkers. The algorithm used to assign a data pixel to a clump of emission can be compared to that of a fell-walker ascending a hill by following the line of steepest ascent (not perhaps the most sensible way to ascend a hill in practise but one which lends some verisimilitude to the present algorithm!).

The algorithm considers every data pixel in the supplied array in turn as a potential start for a "walk" to a neighbouring peak. Pixels which are below a nominated background level (specified by configuration parameter `FellWalker.Noise`) are ignored and are flagged as not belonging to any emission clump. These are skipped over, as are pixels which have already been assigned to a clump. Once a pixel is found that has not yet been assigned to a clump and is above the background level, the algorithm proceeds to trace out a route from this pixel to a neighbouring peak. It does this in a series of steps (comparable to the steps of a fell-walker). At each step the algorithm looks at the 3×3 square of neighbouring pixels (a 3×3×3 cube for 3D data), and selects the neighbour which would produce the highest gradient for the next step. The algorithm then moves to that pixel and proceeds with the next step.

Eventually, this algorithm will reach a local maximum; a point from which all routes go downhill. But this may be merely a noise spike, rather than a significant peak, and so a check is made over a more extended neighbourhood to see if a pixel with a higher data value than the current pixel can be found (the extent of this extended neighbourhood is specified by the configuration parameter `FellWalker.MaxJump`). If so, the algorithm "jumps" to the pixel with the highest value in the extended neighbourhood, and then proceeds as before to walk up-hill. If no pixel with a higher value is found within the extended neighbourhood, the pixel is designated as a significant peak, and is assigned a unique integer identifier. This integer is used to identify all pixels which are within the clump of emission containing the peak, and all pixels which were visited along the walk are assigned this identifier.

If, in the process of walking up-hill, a pixel is visited which has already been assigned to a clump, then the walk is terminated at that point and all the pixels so far visited are assigned to the same clump.

In some cases, the initial part of a walk may be over very flat "terrain". The significant part of a walk is considered to start when the average gradient (taken over a 4 step length) first reaches the value of configuration parameter `FlatSlope`. Any pixels visited prior to this point are deemed not to be in any clump. However, this only applies if the walk starts at or close to "sea level". For walks which start from a higher level (*i.e.* from a pixel which has a data value greater than the selected background level plus twice the RMS noise level), the entire length of the walk is used, including any initial flat section.

Once all pixels in the data array have been considered as potential starts for such a walk, an array will have been created which holds an integer clump identifier for every data pixel. To reduce the effect of noise on the boundaries between clumps, a cellular automata can be used

to smooth the boundaries. This replaces each clump identifier by the most commonly occurring value within a 3×3 square (or 3×3×3 cube for 3D data) of neighbours. The number of times which this cleaning process should be applied is specified by configuration parameter `CleanIter`.

If the high data values in a clump form a plateau with slight undulations, then the above algorithm may create a separate clump for each undulation. This is probably inappropriate, especially if the dips between the undulations are less than or are comparable to the noise level in the data. This situation can arise for instance if the pixel-to-pixel noise is correlated on a scale equal to or larger than the value of the `MaxJump` configuration parameter. To avoid this, adjoining clumps are merged together if the dip between them is less than a specified value. Specifically, if two clumps with peak values *PEAK1* and *PEAK2*, where *PEAK1* is less than *PEAK2*, are adjacent to each other, and if the pixels along the interface between the two clumps all have data values which are larger than "*PEAK1* - *MinDip*" (where *MinDip* is the value of the `MinDip` configuration parameter), then the two clumps are merged together.

The results of this merging process are the final clump allocations for every data pixel, from which the catalogue of clump parameters is produced.

3 Examining the Results

For all algorithms other than GaussClumps, the pixel values in the output NDF created by FINDCLUMPS and EXTRACTCLUMPS are integer values that indicate which clump each pixel belongs to (that is, all pixels that are contain within a single clump will all have the same integer value in the output NDF). Pixels that are not contained within any clump have bad values. For GaussClumps, each pixel value in the output NDF is the sum of the Gaussian models that contribute to that pixel.

3.1 2D Data

For 2D data the simplest way to examine the results is just to display the output NDF using the KAPPA DISPLAY command. For instance:

```
% display clumps mode=scale badcol=red accept
```

will display the file `clumps.sdf` so that black corresponds to the lowest clump index and white to the largest, with bad pixels (*i.e.* pixels not in any clump) coloured red.

If you want to see the actual data values instead of the clump index values, then you can use KAPPA COPYBAD to produce a copy of the original data with all non-clump pixels set bad:

```
% copybad data ref=clumps out=data2
% display data2 mode=perc badcol=red accept
```

will copy the file `data.sdf` into the file `data2.sdf`, setting pixels bad in `data2.sdf` if the corresponding pixels are bad in `clumps.sdf`.

Alternatively, you may be interested in the background (non-clump) pixels. To get an NDF containing just the background pixels, do:

```
% copybad data ref=clumps out=data2 invert
% display data2 mode=perc badcol=red accept
```

The output NDF contains an extension structure holding information about each identified clump. For each clump, the extension contains a minimal cut-out from the input data array that contains just those pixels belonging to the clump[2]. So to display an image of (say) clump 12, do:

```
% display "clumps.more.cupid.clumps(12)" mode=perc badcol=red accept
```

To draw an outline of a clump on top of a previously displayed image of the entire data array, do:

```
% display data mode=perc accept
% outlineclump clumps 12
```

3.2 3D Data

Visualising clumps in 3D is much harder. Options include the following.

1. Examining specified 2D slices from the 3D data. For instance to display the intersection of the tenth pixel plane with clump number 12, do:

   ```
   % display "clumps.more.cupid.clumps(12)(,,10)" mode=perc badcol=red accept
   ```

2. Use the 3D facilities of GAIA (SUN/214), accessed through the `Open cube` entry in the `File` menu.

3. Use a complete 3D visualisation such as OpenDX (see http://www.opendx.org/). The facilities of the Starlink DX extension package (known as "SX" SX (SUN/203)) may be useful as it provides a means of converting NDF data files to the DX native format, and also provides some demonstration DX "networks" that can be used to do simple visualisations of a 3D data cube.

3.3 Using the output Catalogue

By default, the output catalogue created by FINDCLUMPS and EXTRACTCLUMPS will be a standard FITS binary table, with the following columns:

Peak1 : The first co-ordinate at the centre of the pixel with the highest data value in the clump.

Peak2 : The second co-ordinate at the centre of the pixel with the highest data value in the clump.

[2]Any pixels within the bounds of the rectangular region covered by the cut-out that are *not* contained within the clump are set bad.

Peak3 : The third co-ordinate at the centre of the pixel with the highest data value in the clump.

Cen1 : The first co-ordinate of the clump centroid.

Cen2 : The second co-ordinate of the clump centroid.

Cen3 : The third co-ordinate of the clump centroid.

Size1 : The size of the clump along the first axis, in pixels.

Size2 : The size of the clump along the second axis, in pixels.

Size3 : The size of the clump along the third axis, in pixels.

Sum : The total data sum in the clump.

Peak : The peak value in the clump.

Volume : The total number of pixels falling within the clump (a volume for 3D data and an area for 2D data).

The co-ordinate system used depends on the value supplied for the WCSPAR parameter. If WCSPAR is set to TRUE, then current WCS co-ordinate system in the input NDF will be used. If WCSPAR is left at its default value of FALSE, then the input NDF's PIXEL co-ordinate system is used. The centroid position is the weighted mean of the pixel co-ordinate values at the centre of all the pixels in the clump, with the pixel values being used as the weights (the final pixel position will be converted to a WCS position for storage in the catalogue if WCSPAR is TRUE):

$$\begin{aligned} Cen1 &= \frac{\sum_k X_k.D_k}{\sum_k D_k} \\ Cen2 &= \frac{\sum_k Y_k.D_k}{\sum_k D_k} \\ Cen3 &= \frac{\sum_k Z_k.D_k}{\sum_k D_k} \end{aligned}$$

The clump sizes are the standard deviation of the pixel co-ordinate values about the centroid position, weighted by the pixel values. For a Gaussian profile, this size value is equal to the standard deviation of the Gaussian. These sizes are then corrected to remove the effect of the instrumental smoothing specified by the `FwhmBeam` and `VeloRes` configuration parameters (assuming the DECONV parameter is set to TRUE):

$$\begin{aligned} Size1 &= \sqrt{\frac{\sum_k D_k.(X_k - Cen1)^2}{\sum_k D_k} - b_x^2} \\ Size2 &= \sqrt{\frac{\sum_k D_k.(Y_k - Cen2)^2}{\sum_k D_k} - b_y^2} \\ Size3 &= \sqrt{\frac{\sum_k D_k.(Z_k - Cen3)^2}{\sum_k D_k} - b_z^2} \end{aligned}$$

where b_x, b_y and b_z are the beam widths implied by configuration parameters `FwhmBeam` and `VeloRes`. Clumps are excluded from the returned list if the clump size before correction is smaller than the specified beam width. The final widths are converted to WCS units for storage in the catalogue if WCSPAR is TRUE.

If DECONV is TRUE, the peak value in the clump is also corrected to take account of the smoothing produced by the instrumental beam. Such smoothing will result in the observed peak value being less than the real peak value, by an amount that increases as the clump area gets smaller. The correction factor assumes that the clumps has a Gaussian profile and is determined by the requirement that the total data sum within the corrected clump equals the total data sum within the uncorrected clump:

$$Peak \quad = \quad D_{max}.sqrt(Size1'.Size2'.Size3')/(Size1.Size2.Size3)$$

where the primed sizes refer to the clump sizes before the correction for the instrumental beam width.

The recommended way to explore and examine the output catalogue is to use the Starlink catalogue browser, TOPCAT (see http://www.starlink.ac.uk/topcat/).

4 Taking Account of Varying Noise Levels

All the clump finding algorithms implemented by the FINDCLUMPS command assumes that the noise level is constant across the supplied data array, and equal to the the value of the RMS parameter. This is true even if the supplied NDF contains a VARIANCE component[3].

However, in many cases the real noise level may vary across the data array. This may result in real clumps being missed in low noise areas and spurious noise spikes being interpreted as real clumps in high noise areas. To avoid this some way of taking account of the varying noise level is needed. Since the assumption of constant noise level is more or less intrinsic to most of the clump finding algorithms, this is best done by first converting the data array into an array containing the signal-to-noise (SNR) ratio, and then running FINDCLUMPS on this SNR array rather than the original data array. This will determine the spatial extent of each clump, but the output catalogue will contain clump parameters in terms of SNR values rather than the original data values. Therefore, the EXTRACTCLUMPS command should then be used to transfer the clumps outlines found within the SNR array into the original data array and extract the corresponding clump parameters.

So the procedure is as follows.

1. If you have a single NDF containing both DATA and VARIANCE components, use the MAKESNR command (part of the KAPPA package - see SUN/95) to convert the original data array into an SNR array. MAKESNR divides the DATA component of the NDF by the square root of the VARIANCE component, checking for anomalously small variance

[3]Although any available VARIANCE component will be used to determine the default value for the RMS parameter. The GaussClumps algorithm will also use any available variance values to weight the data when fitting individual Gaussians.

values in order to avoid very large spurious SNR values appearing in the output. Any such pixels are assigned a "bad" value in the output SNR array and are excluded from all later calculations.

If you have separate data and noise arrays, then a suitable SNR array can be produced using the KAPPA MATHS command.

The noise level in the SNR array will, by definition, be constant and equal to 1.0.

2. If you wish to apply any smoothing to the array, it should be done now. That is, it is usually better to smooth the SNR array rather than the data array. This is because smoothing the data array can spread anomalous variance values out around the neighbouring pixels, making the anomalous values harder to identify. Such smoothing will not introduce variations in the noise level (assuming the degree of smoothing is constant across the image), but will change the constant noise level from its initial value of 1.0.

3. Use FINDCLUMPS to identify the clumps within the SNR array. The output catalogue will contain clump parameters in terms of SNR value and so will probably not be what you want. However, the output mask NDF will identify the pixels contained in each SNR clump and these will usually correspond to the pixels within each data clump.

4. Use EXTRACTCLUMPS to create a catalogue of clump parameters in terms of the original data value. EXTRACTCLUMPS reads in the mask produced by FINDCLUMPS and identifies the corresponding pixels in the original data array, producing the required data clump parameters.

A Description of the CUPID applications

A.1 Alphabetic list of CUPID routines.

A.2 Complete routine descriptions

The CUPID routine descriptions are contained in the following pages. These descriptions follow the style used in SUN/95 for NDF applications.

CLUMPINFO Obtain information about one or more previously identified clumps CLUMPINFO

Description: This application returns various items of information about a single clump, or a collection of clumps, previously identified using FINDCLUMPS or EXTRACTCLUMPS.

Usage:

```
clumpinfo ndf clumps quiet
```

Parameters:

CLUMPS = LITERAL (Read)
Specifies the indices of the clumps to be included in the returned information. It can take any of the following values:

- "ALL" or "*" – All clumps.
- "xx,yy,zz" – A list of clump indices.
- "xx:yy" – Clump indices between xx and yy inclusively. When xx is omitted the range begins from one; when yy is omitted the range ends with the final clump index.
- Any reasonable combination of above values separated by commas.

FLBND( ) = _DOUBLE (Write)
The lower bounds of the bounding box enclosing the selected clumps in the current WCS Frame of the input NDF. Celestial axis values are always in units of radians, but spectral axis units will be in the spectral units used by the current WCS Frame.

FUBND( ) = _DOUBLE (Write)
The upper bounds of the bounding box enclosing the selected clumps. See parameter FLBND for more details.

LBOUND( ) = _INTEGER (Write)
The lower pixel bounds of bounding box enclosing the selected clumps.

NCLUMPS = _INTEGER (Write)
The total number of clumps descrriptions stored within the supplied NDF.

NDF = NDF (Read)
The NDF defining the previously identified clumps. This should contain a CUPID extension describing all the identified clumps, in the format produced by FINDCLUMPS or EXTRACTCLUMPS.

QUIET = _LOGICAL (Read)
If TRUE, then no information is written out to the screen, although the output parameters are still assigned values. [FALSE]

UBOUND( ) = _INTEGER (Write)
The upper pixel bounds of bounding box enclosing the selected clumps.

Notes:

- It is hoped to extend the range of information reported by this application as new requirements arise.

CUPIDHELP Display information about CUPID CUPIDHELP

Description: This application displays information about CUPID. This includes general topics common to all applications, as well as detailed descriptions of each application. The information is organised in a hierarchical structure of topics, subtopics, sub-subtopics, etc. Each entry ends with a list of related sub-topics. You can then examine any of these sub-topics or return to the previous level.

Usage:

```
cupidhelp [topic] [subtopic] [subsubtopic] [subsubsubtopic]
```

Parameters:

TOPIC = LITERAL (Read)
Topic for which help is to be given. [" "]

SUBTOPIC = LITERAL (Read)
Subtopic for which help is to be given. [" "]

SUBSUBTOPIC = LITERAL (Read)
Subsubtopic for which help is to be given. [" "]

SUBSUBSUBTOPIC = LITERAL (Read)
Subsubsubtopic for which help is to be given. [" "]

Navigating The Help Tree:

The text for each topic is displayed in screen-fulls. A prompt is issued at the end of each topic at which you may:

- enter a topic and/or subtopic name(s) to display the help for that topic or subtopic, so for example, "polka parameters dpi" gives help on DPI, which is a subtopic of Parameters, which in turn is a subtopic of FINDCLUMPS;
- press the RETURN key to see more text at a "Press RETURN to continue ..." request;
- press the RETURN key at topic and subtopic prompts to move up one level in the hierarchy, and if you are at the top level it will terminate the help session;
- enter CTRL/D (i.e. press the CTRL and D keys simultaneously) in response to any prompt will terminate the help session;
- enter a question mark "?" to redisplay the text for the current topic, including the list of topic or subtopic names; or
- enter an ellipsis "..." to display all the text below the current point in the hierarchy. For example, "FINDCLUMPS..." displays information on the FINDCLUMPS topic as well as information on all the subtopics under FINDCLUMPS.

You can abbreviate any topic or subtopic using the following rules.

- Just give the first few characters, e.g. "PARA" for Parameters.

- Some topics are composed of several words separated by underscores. Each word of the keyword may be abbreviated, e.g. "Colour_Set" can be shortened to "C_S".
- The characters "%" and "∗" act as wild-cards, where the percent sign matches any single character, and asterisk matches any sequence of characters. Thus to display information on all available topics, type an asterisk in reply to a prompt.
- If a word contains, but does end with an asterisk wild-card, it must not be truncated.
- The entered string must not contain leading or embedded spaces.

Ambiguous abbreviations result in all matches being displayed.

EXTRACTCLUMPS Extract previously identified clumps of emission from an NDF EXTRACTCLUMPS

Description: This application extract previously identified clumps of emission from a 1, 2 or 3 dimensional NDF. Usually, FINDCLUMPS will first be used to identify the clumps within a given array, and then EXTRACTCLUMPS can be used to find the parameters of the same clumps in a second array.

Two input NDFs are supplied; the NDF associated with parameter DATA contains the physical data values from which the clumps are to be extracted, whilst the NDF associated with parameter MASK contains integer values that identify the clump to which each pixel belongs. The two NDFs are assumed to be aligned in PIXEL coordinates. An output NDF is created that is a copy of the MASK NDF. Parameters describing the clumps extracted from the DATA NDF are stored in the CUPID extension of the output NDF, and may also be stored in an output catalogue. These are in the same form as the clump parameters created by the FINDCLUMPS command.

Usage:

```
extractclumps mask data out outcat
```

Parameters:

BACKOFF = _LOGICAL (Read)
If TRUE, the background level in each clump is removed from the clump data values before calculating the reported clump sizes and centroid position. This means that the clump sizes and centroid position will be independent of the background level. The background level used is the minimum data value in the clump. If FALSE, the full data values, including background, are used when calculating the clump sizes and centroid position. This means that clumps on larger backgrounds will be reported as wider than similar clumps on smaller backgrounds. Using BACKOFF=FALSE allows the comparison of clump properties generated by EXTRACTCLUMPS with those generated by the IDL version of CLUMPFIND (see also the ClumpFind configuration parameter "IDLAlg"). Note, the other reported clump properties such as total data value, peak data value, etc, are always based on the full clump data values, including background. [TRUE]

FWHMBEAM = _REAL (Read)
The FWHM of the instrument beam, in pixels. If DECONV is TRUE, the clump widths written to the output catalogue are reduced (in quadrature) by this amount. The default value is the value stored in the CONFIG component of the CUPID extension in the mask NDF, or 2.0 if the CUPID extension does not contain a CONFIG component. []

DATA = NDF (Read)
The input NDF containing the physical data values.

DECONV = _LOGICAL (Read)
Determines if the clump properties stored in the output catalogue and NDF extension should be corrected to remove the effect of the instrumental beam width specified by

the FWHMBEAM and VELORES parameters. If TRUE, the clump sizes will be reduced and the peak values increased to take account of the smoothing introduced by the beam width. If FALSE, the undeconvolved values are stored in the output catalogue and NDF. Note, the filter to remove clumps smaller than the beam width is still applied, even if DECONV is FALSE. [TRUE]

LOGFILE = LITERAL (Read)
The name of a text log file to create. If a null (!) value is supplied, no log file is created. [!]

MASK = NDF (Read)
The input NDF containing the pixel assignments. This will usually have been created by the FINDCLUMPS command.

OUT = NDF (Write)
The output NDF.

OUTCAT = FILENAME (Write)
An optional output catalogue in which to store the clump parameters. See the description of the OUTCAT parameter for the FINDCLUMPS command for further information.

VELORES = _REAL (Read)
The velocity resolution of the instrument, in channels. If DECONV is TRUE, the velocity width of each clump written to the output catalogue is reduced (in quadrature) by this amount. The default value is the value stored in the CONFIG component of the CUPID extension in the mask NDF, or 2.0 if the CUPID extension does not contain a CONFIG component. []

WCSPAR = _LOGICAL (Read)
If a TRUE value is supplied, then the clump parameters stored in the output catalogue and in the CUPID extension of the output NDF, are stored in WCS units, as defined by the current coordinate frame in the WCS component of the input NDF (this can be inspected using the KAPPA:WCSFRAME command). For instance, if the current coordinate system in the input NDF is (RA,Dec,freq), then the catalogue columns that hold the clump peak and centroid positions will use this same coordinate system. The spatial clump sizes will be stored in arc-seconds, and the spectral clump size will be stored in the unit of frequency used by the NDF (Hz, GHz, etc). If a FALSE value is supplied for this parameter, the clump parameters are stored in units of pixels within the pixel coordinate system of the input NDF. The dynamic default for this parameter is TRUE if the current coordinate system in the input NDF represents celestial longitude and latitude in some system, plus a recogonised spectral axis (if the input NDF is 3D). Otherwise, the dynamic default is FALSE. []

FINDBACK Estimate the background in an NDF by removing small scale structure FINDBACK

Description: This application uses spatial filtering to remove structure with a scale size less than a specified size from a 1, 2, or 3 dimensional NDF, thus producing an estimate of the local background within the NDF.

The algorithm proceeds as follows. A filtered form of the input data is first produced by replacing every input pixel by the minimum of the input values within a rectangular box centred on the pixel. This filtered data is then filtered again, using a filter that replaces every pixel value by the maximum value in a box centred on the pixel. This produces an estimate of the lower envelope of the data, but usually contains unacceptable sharp edges. In addition, this filtered data has a tendency to hug the lower envelope of the noise, thus under-estimating the true background of the noise-free data. The first problem is minimised by smoothing the background estimate using a filter that replaces every pixel value by the mean of the values in a box centred on the pixel. The second problem is minimised by estimating the difference between the input data and the background estimate within regions well removed from any bright areas. This difference is then extrapolated into the bright source regions and used as a correction to the background estimate. Specifically, the residuals between the input data and the initial background estimate are first formed, and residuals which are more than three times the RMS noise are set bad. The remaining residuals are smoothed with a mean filter. This smoothing will replace a lot of the bad values rejected above, but may not remove them all. Any remaining bad values are estimated by linear interpolation between the nearest good values along the first axis. The interpolated residuals are then smoothed again using a mean filter, to get a surface representing the bias in the initial background estimate. This surface is finally added onto the initial background estimate to obtain the output NDF.

Usage:

```
findback in out box
```

Parameters:

BOX() = _INTEGER (Read)

The dimensions of each of the filters, in pixels. Each value should be odd (if an even value is supplied, the next higher odd value will be used). The number of values supplied should not exceed the number of significant (i.e. more than one element) pixel axes in the input array. If any trailing values of 1 are supplied, then each pixel value on the corresponding axes will be fitted independently of its neighbours. For instance, if the data array is 3-dimensional, and the third BOX value is 1, then each x-y plane will be fitted independently of the neighbouring planes. If the NDF has more than 1 pixel axis but only 1 value is supplied, then the same value will be used for the both the first and second pixel axes (a value of 1 will be assumed for the third axis if the input array is 3-dimensional).

MSG_FILTER = _CHAR (Read)

Controls the amount of diagnostic information reported. This is the standard messaging level. The default messaging level is NORM (2). A value of NONE or 0 will suppress all screen output. VERB (3) will indicate progress through the various stages of the algorithm. [NORM]

IN = NDF (Read)
The input NDF.

RMS = _DOUBLE (Read)
Specifies a value to use as the global RMS noise level in the supplied data array. The suggested default value is the square root of the mean of the values in the input NDF's Variance component. If the NDF has no Variance component, the suggested default is based on the differences between neighbouring pixel values, measured over the entire input NDF. If multiple slices within the NDF are to be processed independently (see parameter BOX), it may be more appropriate for a separate default RMS to be calculated for each slice. This will normally be the case if the noise could be different in each of the slices. In such cases a null (!) can be supplied for the RMS parameter, which forces a separate default RMS value to be found and used for each slice. Any pixel-to-pixel correlation in the noise can result in these defaults being too low.

SUB = _LOGICAL (Read)
If a TRUE value is supplied, the output NDF will contain the difference between the supplied input data and the estimated background. If a FALSE value is supplied, the output NDF will contain the estimated background itself. [FALSE]

OUT = NDF (Write)
The output NDF containing either the estimated background, or the background-subtracted input data, as specified by parameter SUB.

Notes:

- Smoothing cubes in 3 dimensions can be very slow.

FINDCLUMPS Identify clumps of emission within a 1, 2 or 3 dimensional NDF FINDCLUMPS

Description: This application identifies clumps of emission within a 1, 2 or 3 dimensional NDF. It is assumed that any background has already been removed from the data array (for instance, using CUPID:FINDBACK). Information about the clumps is returned in several different ways:

- A pixel mask identifying pixels as background, clump or edge pixels is written to the Quality array of each output NDF (see parameters OUT and QOUT). Three quality bits will be used; one is set if and only if the pixel is contained within one or more clumps, another is set if and only if the pixel is not contained within any clump, and the other is set if and only if the pixel is in a clump but on the edge of the clump (i.e. has one or more neighbouring pixels that are not inside a clump). These three quality bits have names associated with them which can be used with the KAPPA applications SETQUAL, QUALTOBAD, REMQUAL, SHOWQUAL. The names used are "CLUMP", "BACKGROUND" and "EDGE". For instance, to overlay the outline of a set of 2D clumps held in NDF "fred" on a previously displayed 2D image, do "qualtobad fred fred2 background" followed by "contour noclear mode=good fred2".
- Information about each clump, including a minimal cut-out image of the clump and the clump parameters, is written to the CUPID extension of the output NDF (see the section "Use of CUPID Extension" below).
- An output catalogue containing clump parameters can be created (see parameter OUTCAT).

The algorithm used to identify the clumps (GaussCLumps, ClumpFind, etc) can be specified (see parameter METHOD).

Usage:

```
findclumps in out outcat method
```

Parameters:

BACKOFF = _LOGICAL (Read)

If TRUE, the background level in each clump is removed from the clump data values before calculating the reported clump sizes and centroid position (the background level used is the minimum data value in the clump). If FALSE, the full data values, including background, are used when calculating the clump sizes and centroid position.

If BACKOFF is FALSE, a clump that sits on a high background level will have a larger reported width than an identical clump sitting on a lower background level. The position of the centroid may also be affected by the background level. This is usually undesirable, and so the default value for BACKOFF is usually TRUE. The main reason you may want to set BACKOFF to FALSE is if you want to compare clump properties found by FINDCLUMPS with those found by the IDL version of CLUMPFIND

(which includes the background in its calculations). For this reason, the dynamic default value got BACKOFF is TRUE, unless METHOD is "ClumpFind" and the ClumpFind.IDLAlg configuration parameter is non-zero, in which case the dynamic default for BACKOFF is FALSE.

Note, the other reported clump properties such as total data value, peak data value, etc, are always based on the full clump data values, including background. []

CONFIG = GROUP (Read)

Specifies values for the configuration parameters used by the clump finding algorithms. If the string "def" (case-insensitive) or a null (!) value is supplied, a set of default configuration parameter values will be used.

The supplied value should be either a comma-separated list of strings or the name of a text file preceded by an up-arrow character "^", containing one or more comma-separated list of strings. Each string is either a "keyword=value" setting, or the name of a text file preceded by an up-arrow character "^". Such text files should contain further comma-separated lists which will be read and interpreted in the same manner (any blank lines or lines beginning with "#" are ignored). Within a text file, newlines can be used as delimiters as well as commas. Settings are applied in the order in which they occur within the list, with later settings over-riding any earlier settings given for the same keyword.

Each individual setting should be of the form:

<keyword>=<value>

where <keyword> has the form "algorithm.param"; that is, the name of the algorithm, followed by a dot, followed by the name of the parameter to be set. If the algorithm name is omitted, the current algorithm given by parameter METHOD is assumed. The parameters available for each algorithm are listed in the "Configuration Parameters" sections below. Default values will be used for any unspecified parameters. Assigning the value "<def>" (case insensitive) to a keyword has the effect of reseting it to its default value. Unrecognised options are ignored (that is, no error is reported). [current value]

DECONV = _LOGICAL (Read)

Determines if the clump properties stored in the output catalogue and NDF extension should be corrected to remove the effect of the instrumental beam width specified by the FwhmBeam and VeloRes configuration parameters. If TRUE, the clump sizes will be reduced and the peak values increased to take account of the smoothing introduced by the beam width. If FALSE, the undeconvolved values are stored in the output catalogue and NDF. Note, the filter to remove clumps smaller than the beam width is still applied, even if DECONV is FALSE. [TRUE]

MSG_FILTER = _CHAR (Read)

Controls the amount of diagnostic information reported. It uses the standard message filtering system. It should be in the range 0 to 6 (NONE, QUIET, NORM, VERB, DEBUG, DEBUG1-3). A value of NONE (zero) will suppress all screen output. Larger values give more information (the precise information displayed depends on the algorithm being used). Note, this screen output describes the progress of the specific clump finding algorithm selected using the METHOD parameter, and therefore clump parameters such as clump size, etc, will be displayed using the definition most natural to the chosen algorithm. These definitions may not be the same as those used to create the output catalogue, since the output catalogue contains standardised columns

chosen to allow comparison between different algorithms. For instance, the clump sizes displayed on the screen by the GaussClumps algorithm will be FWHM in pixels, but the clump sizes stored in the output catalogue are the RMS deviation of each pixel centre from the clump centroid, weighted by the corresponding pixel data value. [NORM]

IN = NDF (Read)
The 1, 2 or 3 dimensional NDF to be analysed.

LOGFILE = LITERAL (Read)
The name of a text log file to create. If a null (!) value is supplied, no log file is created. [!]

METHOD = LITERAL (Read)
The algorithm to use. Each algorithm is described in more detail in the "Algorithms:" section below. Can be one of:

- GaussClumps
- ClumpFind
- Reinhold
- FellWalker

Each algorithm has a collection of extra tuning values which are set via the CONFIG parameter. [current value]

NCLUMPS = _INTEGER (Write)
The total number of clumps descrriptions stored within the output NDF (and catalogue).

OUT = NDF (Write)
The output NDF which has the same shape and size as the input NDF. Information about the identified clumps and the configuration parameters used will be stored in the CUPID extension of this NDF. See "Use of CUPID Extension" below for further details about the information stored in the CUPID extension. Other applications within the CUPID package can be used to display this information in various ways. The information written to the DATA array of this NDF depends on the value of the METHOD parameter. If METHOD is GaussClumps, the output NDF receives the sum of all the fitted Gaussian clump models including a global background level chosen to make the mean output value equal to the mean input value. If METHOD is ClumpFind, FellWalker or Reinhold, each pixel in the output is the integer index of the clump to which the pixel has been assigned. Bad values are stored for pixels which are not part of any clump. The output NDF will inherit the AXIS and WCS components (plus any extensions) from the input NDF.

OUTCAT = FILENAME (Write)
An optional output catalogue in which to store the clump parameters. No catalogue will be produced if a null (!) value is supplied. The following columns are included in the output catalogue:

- Peak1: The position of the clump peak value on axis 1.
- Peak2: The position of the clump peak value on axis 2.
- Peak3: The position of the clump peak value on axis 3.

- Cen1: The position of the clump centroid on axis 1.
- Cen2: The position of the clump centroid on axis 2.
- Cen3: The position of the clump centroid on axis 3.
- Size1: The size of the clump along pixel axis 1.
- Size2: The size of the clump along pixel axis 2.
- Size3: The size of the clump along pixel axis 3.
- Sum: The total data sum in the clump.
- Peak: The peak value in the clump.
- Volume: The total number of pixels falling within the clump.

There is also an optional column called "Shape" containing an STC-S description of the spatial coverage of each clump. See parameter SHAPE.

The coordinate system used to describe the peak and centroid positions is determined by the value supplied for parameter WCSPAR. If WCSPAR is FALSE, then positions are specified in the pixel coordinate system of the input NDF. In addition, the clump sizes are specified in units of pixels, and the clump volume is specified in units of cubic pixels (square pixels for 2D data). If WCSPAR is TRUE, then positions are specified in the current coordinate system of the input NDF. In addition, the clump sizes and volumes are specified in WCS units. Note, the sizes are still measured parallel to the pixel axes, but are recorded in WCS units rather than pixel units. Celestial coordinate positions are units of degrees, sizes are in units are arc-seconds, and areas in square arc-seconds. Spectral coordinates are in the units displayed by the KAPPA command "ndftrace".

If the data has less than 3 pixel axes, then the columns describing the missing axes will not be present in the catalogue.

The catalogue inherits any WCS information from the input NDF.

The "size" of the clump on an axis is the RMS deviation of each pixel centre from the clump centroid, where each pixel is weighted by the corresponding pixel data value. For a Gaussian profile, this "size" value is equal to the standard deviation of the Gaussian. Optionally, the weights can be be based on the pixel data value after removal of the background - see parameter BACKOFF). If parameter DECONV is set TRUE, the values stored for "Size..." and "Peak" are corrected to take account of the smoothing introduced by the instrumental beam. These corrections reduced the "size..." values and increase the peak value. Beam sizes are specified by configuration parameters FWHMBeam and VeloRes.

For the GaussClump algorithm, the Sum and Volume values refer to the part of the Gaussian within the level defined by the GaussClump.ModelLim configuration parameter.

The values used for configuration parameters and ADAM parameters are written to the history information of the output catalogue.

The KAPPA command "listshow" can be used to draw markers at the central positions of the clumps described in a catalogue. For instance, the command "listshow fred plot=mark" will draw markers identifying the positions of the clumps described in file fred.FIT, overlaying the markers on top of the currently displayed image. Specifying "plot=STCS" instead of "plot=mark" will cause the spatial outline of the clump to be drawn if it is present in the catalogue (see parameter SHAPE). [!]

PERSPECTRUM = _LOGICAL (Read)
This parameter is ignored unless the supplied input NDF is 3-dimensional and includes a spectral axis. If so, then a TRUE value for PERSPECTRUM will cause all spectra within the supplied cube to be processed independenly of the neighbouring spectra. That is, each identified clump will contain pixels from only a single input spectrum. If a clump extends across multiple spectra, then it will be split up into multiple clumps, one for each spectrum. Currently, this parameter can only be used with the FellWalker and ClumpFind methods. A value of FALSE is always used for other methods. [FALSE]

QOUT = NDF (Write)
An optional output NDF that is a copy of the input NDF, except that any Quality component in the input NDF is discarded and a new one created. The new Quality component defines 3 flags that indicate if each pixel is inside a clump, on the edge of a clump or outside all clumps. If a null (!) value is supplied, no NDF is created. [!]

REPCONF = _LOGICAL (Read)
If a TRUE value is supplied, then the configuration parameters supplied by the CONFIG parameter will be listed to standard output. [current value]

RMS = _DOUBLE (Read)
Specifies a value to use as the global RMS noise level in the supplied data array. The suggested default value is the square root of the mean of the values in the input NDF's Variance component. If the NDF has no Variance component, the suggested default is based on the differences between neighbouring pixel values. Any pixel-to-pixel correlation in the noise can result in this estimate being too low. The value supplied for this parameter will be ignored if the RMS noise level is also given in the configuration file specified by parameter CONFIG.

SHAPE = LITERAL (Read)
Specifies the shape that should be used to describe the spatial coverage of each clump in the output catalogue. It can be set to "None", "Polygon" or "Ellipse". If it is set to "None", the spatial shape of each clump is not recorded in the output catalogue. Otherwise, the catalogue will have an extra column named "Shape" holding an STC-S description of the spatial coverage of each clump. "STC-S" is a textual format developed by the IVOA for describing regions within a WCS - see http://www.ivoa.net/Documents/latest/STC-S.html for details. These STC-S desriptions can be displayed by the KAPPA:LISTSHOW command, or using GAIA. Since STC-S cannot describe regions within a pixel array, it is necessary to set parameter WCSPAR to TRUE if using this option. An error will be reported if WCSPAR is FALSE. An error will also be reported if the WCS in the input data does not contain a pair of scelestial sky axes.

- Polygon: Each polygon will have, at most, 15 vertices. If the data is 2-dimensional, the polygon is a fit to the clump's outer boundary (the region containing all godo data values). If the data is 3-dimensional, the spatial footprint of each clump is determined by rejecting the least significant 10% of spatial pixels, where "significance" is measured by the number of spectral channels that contribute to the spatial pixel. The polygon is then a fit to the outer boundary of the remaining spatial pixels.

- Ellipse: All data values in the clump are projected onto the spatial plane and "size" of the collapsed clump at four different position angles - all separated by 45 degrees - is found (see the OUTCAT parameter for a description of clump "size"). The ellipse that generates the same sizes at the four position angles is then found and used as the clump shape.

In general, "Ellipse" will outline the brighter, inner regions of each clump, and "Polygon" will include the fainter outer regions. ["None"]

WCSPAR = _LOGICAL (Read)

If a TRUE value is supplied, then the clump parameters stored in the output catalogue and in the CUPID extension of the output NDF, are stored in WCS units, as defined by the current coordinate frame in the WCS component of the input NDF (this can be inspected using the KAPPA:WCSFRAME command). For instance, if the current coordinate system in the 3D input NDF is (RA,Dec,freq), then the catalogue columns that hold the clump peak and centroid positions will use this same coordinate system. The spatial clump sizes will be stored in arc-seconds, and the spectral clump size will be stored in the unit of frequency used by the NDF (Hz, GHz, etc). If a FALSE value is supplied for this parameter, the clump parameters are stored in units of pixels within the pixel coordinate system of the input NDF. The dynamic default for this parameter is TRUE if the current coordinate system in the input NDF represents celestial longitude and latitude in some system, plus a recogonised spectral axis (if the input NDF is 3D). Otherwise, the dynamic default is FALSE. []

Use of CUPID Extension:

This application will create an NDF extension called "CUPID" in the output NDF and will add the following components to it:

- CLUMPS: This a an array of CLUMP structures, one for each clump identified by the selected algorithm. Each such structure contains the same clump parameters that are written to the catalogue via parameter OUTCAT. It also contains a component called MODEL which is an NDF containing a section of the main input NDF which is just large enough to encompass the clump. Any pixels within this section which are not contained within the clump are set bad. So for instance, if the input array "fred.sdf" is 2-dimensional, and an image of it has been displayed using KAPPA:DISPLAY, then the outline of clump number 9 (say) in the output image "fred2.sdf" can be overlayed on the image by doing:

contour noclear "fred2.more.cupid.clumps(9).model" mode=good labpos=\!

- CONFIG: Lists the algorithm configuration parameters used to identify the clumps (see parameter CONFIG).
- QUALITY_NAMES: Defines the textual names used to identify background and clump pixels within the Quality mask.

Algorithms:

- GaussClumps: Based on the algorithm described by Stutski & Gusten (1990, ApJ 356, 513). This algorithm proceeds by fitting a Gaussian profile to the brightest peak in the data. It then subtracts the fit from the data and iterates, fitting a new ellipse to the brightest peak in the residuals. This continues until the integrated data sum in the fitted Gaussians reaches the integrated data sum in the input array, or a series of consecutive fits are made which have peak values below a given multiple of the noise level. Each fitted ellipse is taken to be a single clump and is added to the output catalogue. In this algorithm, clumps may overlap. Any input variance component is used to scale the weight associated with each pixel value when performing the Gaussian fit. The most significant configuration parameters for this algorithm are: GaussClumps.FwhmBeam and GaussClumps.VeloRes which determine the minimum clump size.
- ClumpFind: Described by Williams et al (1994, ApJ 428, 693). This algorithm works by first contouring the data at a multiple of the noise, then searches for peaks of emission which locate the clumps, and then follows them down to lower intensities. No a priori clump profile is assumed. In this algorithm, clumps never overlap. Clumps which touch an edge of the data array are not included in the final list of clumps.
- Reinhold: Based on an algorithm developed by Kim Reinhold at JAC. See SUN/255 for more information on this algorithm. The edges of the clumps are first found by searching for peaks within a set of 1D profiles running through the data, and then following the wings of each peak down to the noise level or to a local minimum. A mask is thus produced in which the edges of the clumps are marked. These edges however tend to be quite noisy, and so need to be cleaned up before further use. This is done using a pair of cellular automata which first dilate the edge regions and then erode them. The volume between the edges are then filled with an index value associated with the peak position. Another cellular automata is used to removed noise from the filled clumps.
- FellWalker: Based on an algorithm which walks up hill along the line of greatest gradient until a significant peak is reached. It then assigns all pixels visited along the route to the clump associated with the peak. Such a walk is performed for every pixel in the data array which is above a specified background level. See SUN/255 for more information on this algorithm.

GaussClumps Configuration Parameters :

GaussClumps.FwhmBeam:
The FWHM of the instrument beam, in pixels. The fitted Gaussians are not allowed to be smaller than the instrument beam. This prevents noise spikes being fitted. [2.0]

GaussClumps.FwhmStart:
An initial guess at the ratio of the typical observed clump size to the instrument beam width. This is used to determine the starting point for the algorithm which finds the best fitting Gaussian for each clump. If no value is supplied (or if FwhmBeam is zero), the initial guess at the clump size is based on the local profile around the pixel with peak value. []

GaussClumps.MaxBad:
The maximum fraction of bad pixels which may be included in a clump. Clumps will be excluded if they contain more bad pixels than this value [0.05]

GaussClumps.MaxClumps:
Specifies a termination criterion for the GaussClumps algorithm. The algorithm will terminate when "MaxClumps" clumps have been identified, or when one of the other termination criteria is met. [unlimited]

GaussClumps.MaxNF:
The maximum number of evaluations of the objective function allowed when fitting an individual clump. Here, the objective function evaluates the chi-squared between the current gaussian model and the data being fitted. [100]

GaussClumps.MaxSkip:
The maximum number of consecutive failures which are allowed when fitting Guassians. If more than "MaxSkip" consecutive clumps cannot be fitted, the iterative fitting process is terminated. [10]

GaussClumps.ModelLim:
Determines the value at which each Gaussian model is truncated to zero. Model values below ModelLim times the RMS noise are treated as zero. [0.5]

GaussClumps.NPad:
Specifies a termination criterion for the GaussClumps algorithm. The algorithm will terminate when "Npad" consecutive clumps have been fitted all of which have peak values less than the threshold value specified by the "Thresh" parameter, or when one of the other termination criteria is met. [10]

GaussClumps.RMS:
The global RMS noise level in the data. The default value is the value supplied for parameter RMS. []

GaussClumps.S0:
The Chi-square stiffness parameter "S0" which encourages the fitted gaussian value to be below the corresponding value in the observed data at every point (see the Stutski & Gusten paper). [1.0]

GaussClumps.Sa:
The Chi-square stiffness parameter "Sa" which encourages the peak amplitude of each fitted gaussian to be close to the corresponding maximum value in the observed data (see the Stutski & Gusten paper). [1.0]

GaussClumps.Sb:
An additional Chi-square stiffness parameter which encourages the background value to stay close to its initial value. This stiffness is not present in the Stutzki & Gusten paper but is added because the background value is usually determined by data points which have very low weight and is thus poorly constrained. It would thus be possibly to get erroneous background values without this extra stiffness. [0.1]

GaussClumps.Sc:
The Chi-square stiffness parameter "Sc" which encourages the peak position of each fitted gaussian to be close to the corresponding peak position in the observed data (see the Stutski & Gusten paper). [1.0]

GaussClumps.Thresh:
Gives the minimum peak amplitude of clumps to be fitted by the GaussClumps algorithm (see also GaussClumps.NPad). The supplied value is multipled by the RMS noise level before being used. [2.0]

GaussClumps.VeloRes:
The velocity resolution of the instrument, in channels. The velocity FWHM of each clump is not allowed to be smaller than this value. Only used for 3D data. [2.0]

GaussClumps.VeloStart:
An initial guess at the ratio of the typical observed clump velocity width to the velocity resolution. This is used to determine the starting point for the algorithm which finds the best fitting Gaussian for each clump. If no value is supplied (or if VeloRes is zero), the initial guess at the clump velocity width is based on the local profile around the pixel with peak value. Only used for 3D data. []

GaussClumps.Wmin:
This parameter, together with GaussClumps.Wwidth, determines which input data values are used when fitting a Gaussian to a given peak in the data array. It specifies the minimum weight which is to be used (normalised to a maximum weight value of 1.0). Pixels with weight smaller than this value are not included in the fitting process. [0.05]

GaussClumps.Wwidth:
This parameter, together with GaussClumps.Wmin, determines which input data values are used when fitting a Gaussian to a given peak in the data array. It is the ratio of the width of the Gaussian weighting function (used to weight the data around each clump during the fitting process), to the width of the initial guess Guassian used as the starting point for the Gaussian fitting process. The Gaussian weighting function has the same centre as the initial guess Gaussian. [2.0]

ClumpFind Configuration Parameters :

ClumpFind.AllowEdge:
If set to a zero value, then clumps are rejected if they touch any edge of the data array. If non-zero, then such clumps are retained. Note, other implementations of ClumpFind often include such clumps but flag them in some way. [0]

ClumpFind.DeltaT:
The gap between the contour levels. Only accessed if no value is supplied for "Level1", in which case the contour levels are linearly spaced, starting at a lowest level given by "Tlow" and spaced by "DeltaT". Note, small values of DeltaT can result in noise spikes being interpreted as real peaks, whilst large values can result in some real peaks being missed and merged in with neighbouring peaks. The default value of two times the RMS noise level is usually considered to be optimal, although this obviously depends on the RMS noise level being correct. The value can be supplied either as an absolute data value, or as a multiple of the RMS noise using the syntax "[x]∗RMS", where "[x]" is a numerical value (e.g. "3.2∗RMS"). [2∗RMS]

ClumpFind.FwhmBeam:
The FWHM of the instrument beam, in pixels. If application paremeter DECONV is set TRUE, the clump widths written to the output catalogue are reduced (in quadrature) by this amount. If a direct comparison with other implementations of the ClumpFind algorithm is required, DECONV should be set to FALSE. [2.0]

ClumpFind.IDLAlg:
If a non-zero value is supplied, then FINDCLUMPS emulates the ClumpFind algorithm as implemented by the IDL package available from Jonathan Williams WWW

site on 28th April 2006. The default value of zero causes FINDCLUMPS to use the algorithm described in the Williams et al ApJ paper of 1994. These two algorithms differ in the way that pixels within merged clumps are allocated to individual clumps. Also the ApJ algorithm rejects clumps that do not not extend above the second contour level, whereas the IDL algorithm accepts such clumps. See also parameter BACKOFF. [0]

ClumpFind.Level<n>:
The n'th data value at which to contour the data array (where <n> is an integer). Values should be given for "Level1", "Level2", "Level3", etc. Any number of contours can be supplied, but there must be no gaps in the progression of values for <n>. The values will be sorted into descending order before being used. If "Level1" is not supplied (the default), then contour levels are instead determined automatically using parameters "Tlow" and "DeltaT". Note clumps found at higher contour levels are traced down to the lowest supplied contour level, but any new clumps which are initially found at the lowest contour level are ignored. That is, clumps must have peaks which exceed the second lowest contour level to be included in the returned catalogue. The values can be supplied either as absolute data values, or as mutliples of the RMS noise using the syntax "[x]∗RMS", where "[x]" is a numerical value (e.g. "3.2∗RMS").[]

ClumpFind.MaxBad:
The maximum fraction of pixels in a clump that are allowed to be adjacent to a bad pixel. If the fraction of clump pixels adjacent to a bad pixel exceeds this value, the clump is excluded. If a direct comparison with other implementations of the ClumpFind algorithm is required, a value of 1.0 should be used. [0.05]

ClumpFind.MinPix:
The lowest number of pixel which a clump can contain. If a candidate clump has fewer than this number of pixels, it will be ignored. This prevents noise spikes from being interpreted as real clumps. The default value is based on the supplied values for the other parameters that specify the minimum peak height, the background level and the instrumental beam widths, limited to be at least 16 pixels (for 3D data), 7 pixels (for 2D data) or 3 pixels (for 1D data, or if "PERSPECTRUM" is set TRUE). If a direct comparison with other implementations of the ClumpFind algorithm is required, a value of 5 should be used (for 3D data) or 20 (for 2D data). []

ClumpFind.Naxis:
Controls the way in which contiguous areas of pixels are located when contouring the data. When a pixel is found to be at or above a contour level, the adjacent pixels are also checked. "Naxis" defines what is meant by an "adjacent" pixel in this sense. The supplied value must be at least 1 and must not exceed the number of pixel axes in the data. The default value equals the number of pixel axes in the data. If the data is 3-dimensional, any given pixel can be considered to be at the centre of a cube of neighbouring pixels. If "Naxis" is 1 only those pixels which are at the centres of the cube faces are considered to be adjacent to the central pixel. If "Naxis" is 2, pixels which are at the centre of any edge of the cube are also considered to be adjacent to the central pixel. If "Naxis" is 3, pixels which are at the corners of the cube are also considered to be adjacent to the central pixel. If the data is 2-dimensional, any given pixel can be considered to be at the centre of a square of neighbouring pixels. If "Naxis" is 1 only those pixels which are at the centres of the square edges

are considered to be adjacent to the central pixel. If "Naxis" is 2, pixels which are at square corners are also considered to be adjacent to the central pixel. For one dimensional data, a value of 1 is always used for "Naxis", and each pixel simply has 2 adjacent pixels, one on either side. Note, the supplied "naxis" value is ignored if the ADAM parameter "PERSPECTRUM" is set TRUE. []

ClumpFind.RMS:
The global RMS noise level in the data. The default value is the value supplied for parameter RMS. []

ClumpFind.Tlow:
The lowest level at which to contour the data array. Only accessed if no value is supplied for "Level1". See also "DeltaT". The value can be supplied either as an absolute data value, or as a mutliple of the RMS noise using the syntax "[x]∗RMS", where "[x]" is a numerical value (e.g. "3.2∗RMS"). [2∗RMS]

ClumpFind.VeloRes:
The velocity resolution of the instrument, in channels. The velocity width of each clump written to the output catalogue is reduced (in quadrature) by this amount. If a direct comparison with other implementations of the ClumpFind algorithm is required, a value of zero should be used. [2.0]

Reinhold Configuration Parameters :

Reinhold.FwhmBeam:
The FWHM of the instrument beam, in pixels. If application paremeter DECONV is set TRUE, the clump widths written to the output catalogue are reduced (in quadrature) by this amount. [2.0]

ReinholdClumps.MaxBad:
The maximum fraction of pixels in a clump that are allowed to be adjacent to a bad pixel. If the fraction of clump pixels adjacent to a bad pixel exceeds this value, the clump is excluded. [0.05]

Reinhold.MinLen:
The minimum number of pixels spanned by a peak along any one dimension in order for the peak to be considered significant. If a peak is spanned by fewer than this number of pixels on any axis, then it is ignored. [4]

Reinhold.MinPix:
The lowest number of pixel which a clump can contain. If a candidate clump has fewer than this number of pixels, it will be ignored. This prevents noise spikes from being interpreted as real clumps. The default value is based on the supplied values for the other parameters that specify the minimum peak height, the background level and the instrumental beam widths, limited to be at least 16 pixels(for 3D data), 7 pixels (for 2D data) or 3 pixels (for 1D data). []

Reinhold.Noise:
Defines the data value below which pixels are considered to be in the noise. A peak is considered to end when the peak value dips below the "noise" value. The value can be supplied either as an absolute data value, or as a mutliple of the RMS noise using the syntax "[x]∗RMS", where "[x]" is a numerical value (e.g. "3.2∗RMS"). [2∗RMS]

Reinhold.Thresh:
The smallest significant peak height. Peaks which have a maximum data value less

than this value are ignored. The value can be supplied either as an absolute data value, or as a mutliple of the RMS noise using the syntax "[x]∗RMS", where "[x]" is a numerical value (e.g. "3.2∗RMS"). [Noise+2∗RMS]

Reinhold.FlatSlope:
A peak is considered to end when the slope of a profile through the peak drops below this value. The value should be given as a change in data value between adjacent pixels. The value can be supplied either as an absolute data value, or as a mutliple of the RMS noise using the syntax "[x]∗RMS", where "[x]" is a numerical value (e.g. "3.2∗RMS"). [1.0∗RMS]

Reinhold.CAThresh:
Controls the operation of the cellular automata which is used to erode the (previously dilated) edges regions prior to filling them with clump indicies. If the number of edge pixels in the 3x3x3 pixel cube (or 2x2 pixel square for 2D data) surrounding any pixel is greater than CAThresh, then the central pixel is considered to be an edge pixel. Otherwise it is not considered to be an edge pixel. The default value is one less than the total number of pixels in the square or cube (i.e. 8 for 2D data and 26 for 3D data). []

Reinhold.CAIterations:
This gives the number of times to apply the cellular automata which is used to erode the edges regions prior to filling them with clump indicies. [1]

Reinhold.FixClumpsIterations:
This gives the number of times to apply the cellular automata which cleans up the filled clumps. This cellular automata replaces each output pixel by the most commonly occuring value within a 3x3x3 cube (or 2x2 square for 2D data) of input pixels centred on the output pixel. [1]

Reinhold.RMS:
The global RMS noise level in the data. The default value is the value supplied for parameter RMS. []

Reinhold.VeloRes:
The velocity resolution of the instrument, in channels. The velocity width of each clump written to the output catalogue is reduced (in quadrature) by this amount. [2.0]

FellWalker Configuration Parameters :

FellWalker.AllowEdge:
If set to a zero value, then clumps are rejected if they touch any edge of the data array. If non-zero, then such clumps are retained. [1]

FellWalker.CleanIter:
This gives the number of times to apply the cellular automata which cleans up the filled clumps. This cellular automata replaces each clump index by the most commonly occuring value within a 3x3x3 cube (or 2x2 square for 2D data) of neighbours. The supplied value is ignored and a value of zero is assumed if "PERSPECTRUM" is set TRUE. [1]

FellWalker.FlatSlope:
Any initial section to a walk which has an average gradient (measured over 4 steps) less than this value will not be included in the clump. The value is the data increment

between pixels, and can be supplied either as an absolute data value, or as a mutliple of the RMS noise using the syntax "[x]∗RMS", where "[x]" is a numerical value (e.g. "3.2∗RMS"). [1.0∗RMS]

FellWalker.FwhmBeam:
The FWHM of the instrument beam, in pixels. If application paremeter DECONV is set TRUE, the clump widths written to the output catalogue are reduced (in quadrature) by this amount. [2.0]

FellWalker.MaxBad:
The maximum fraction of pixels in a clump that are allowed to be adjacent to a bad pixel. If the fraction of clump pixels adjacent to a bad pixel exceeds this value, the clump is excluded. [0.05]

FellWalker.MinDip:
If the dip between two adjacent peaks is less than this value, then the peaks are considered to be part of the same clump. The value can be supplied either as an absolute data value, or as a mutliple of the RMS noise using the syntax "[x]∗RMS", where "[x]" is a numerical value (e.g. "3.2∗RMS"). [3.0∗RMS]

FellWalker.MinHeight:
If the peak value in a clump is less than this value then the clump is not included in the returned list of clumps. The value can be supplied either as an absolute data value, or as a mutliple of the RMS noise using the syntax "[x]∗RMS", where "[x]" is a numerical value (e.g. "3.2∗RMS"). [MinDip+Noise]

FellWalker.MinPix:
The lowest number of pixel which a clump can contain. If a candidate clump has fewer than this number of pixels, it will be ignored. This prevents noise spikes from being interpreted as real clumps. The default value is based on the supplied values for the other parameters that specify the minimum peak height, the background level and the instrumental beam widths, limited to be at least 16 pixels (for 3D data), 7 pixels (for 2D data) or 3 pixels (for 1D data). []

FellWalker.MaxJump:
Defines the extent of the neighbourhood about a local maximum which is checked for higher pixel values. The neighbourhood checked is square or cube with side equal to twice the supplied value, in pixels. [4]

FellWalker.Noise:
Defines the data value below which pixels are considered to be in the noise. No walk will start from a pixel with data value less than this value. The value can be supplied either as an absolute data value, or as a mutliple of the RMS noise using the syntax "[x]∗RMS", where "[x]" is a numerical value (e.g. "3.2∗RMS"). [2∗RMS]

FellWalker.RMS:
The global RMS noise level in the data. The default value is the value supplied for parameter RMS. []

FellWalker.VeloRes:
The velocity resolution of the instrument, in channels. The velocity width of each clump written to the output catalogue is reduced (in quadrature) by this amount. [2.0]

MAKECLUMPS Create simulated data containing clumps and noise MAKECLUMPS

Description: This application creates a new 1-, 2- or 3-dimensional NDF containing a collection of clumps with background noise. It also creates a catalogue containing the clump parameters.

The clumps profiles are Gaussian, with elliptical isophotes. The values of each parameter defining the clump shape can be either fixed at a constant value or selected from a given probability distribution.

Usage:

```
makeclumps out outcat
```

Parameters:

ANGLE(2) = _REAL (Read)
Defines the distribution from which the spatial position angle of the major axis of the elliptical clump is chosen. Values should be supplied in units of degrees. See parameter PARDIST for additional information. Note, angles are always taken from a uniform distribution, irrespective of the setting of PARDIST. [current value]

BEAMFWHM = _REAL (Read)
The spatial FHWM (Full Width at Half Max) of the instrumental beam, in pixels. The generated clumps are smoothed with a Gaussian beam of this FWHM, before noise is added. No spatial smoothing is performed if BEAMFWHM is zero. [current value]

DECONV = _LOGICAL (Read)
If TRUE, the clump properties stored in the output catalogue will be modified to take account of the smoothing caused by the instrumental beam width. [TRUE]

FWHM1(2) = _REAL (Read)
Defines the distribution from which the FWHM (Full Width at Half Max) for pixel axis 1 of each clump is chosen. Values should be supplied in units of pixel. See parameter PARDIST for additional information. [current value]

FWHM2(2) = _REAL (Read)
Defines the distribution from which the FWHM (Full Width at Half Max) for pixel axis 2 of each clump is chosen. Values should be supplied in units of pixel. See parameter PARDIST for additional information. [current value]

FWHM3(2) = _REAL (Read)
Defines the distribution from which the FWHM (Full Width at Half Max) for pixel axis 3 of each clump is chosen. Values should be supplied in units of pixel. See parameter PARDIST for additional information. [current value]

LBND() = _INTEGER (Read)
The lower pixel bounds of the output NDF. The number of values supplied (1, 2 or 3) defines the number of pixel axes in the output NDF (an error is reported if the number of values supplied for LBND and UBND differ). If an NDF is supplied for parameter LIKE, the suggested defaults for this parameter will be the lower pixel bounds of the supplied NDF.

LIKE = NDF (Read)
An NDF from which to inherited WCS information. If a null (!) value is supplied, the output catalogue will hold values in pixel coordinates, and there will be no WCS in any of the output NDFs. [!]

NCLUMP = _INTEGER (Read)
The number of clumps to create.

OUT = NDF (Write)
The NDF to receive the simulated data, including instrumental blurring and noise.

MODEL = NDF (Write)
The NDF to receive the simulated data, excluding noise. A CUPID extension is added to this NDF, containing information about each clump in the same format as produced by the FINDCLUMPS command. This includes an NDF holding an of the individual clump.

OUTCAT = FILENAME (Write)
The output catalogue in which to store the clump parameters. There will be one row per clump, with the following columns:

- Peak1: The position of the clump peak value on axis 1.
- Peak2: The position of the clump peak value on axis 2.
- Peak3: The position of the clump peak value on axis 3.
- Cen1: The position of the clump centroid on axis 1.
- Cen2: The position of the clump centroid on axis 2.
- Cen3: The position of the clump centroid on axis 3.
- Size1: The size of the clump along pixel axis 1.
- Size2: The size of the clump along pixel axis 2.
- Size3: The size of the clump along pixel axis 3.
- Sum: The total data sum in the clump.
- Peak: The peak value in the clump.
- Volume: The total number of pixels falling within the clump.

There is also an optional column called "Shape" containing an STC-S description of the spatial coverage of each clump. See parameter SHAPE.

The coordinate system used to describe the peak and centroid positions is determined by the value supplied for parameter LIKE. If LIKE is null (!), then positions are specified in the pixel coordinate system. In addition, the clump sizes are specified in units of pixels, and the clump volume is specified in units of cubic pixels (square pixels for 2D data). If an NDF is supplied for LIKE, then positions are specified in the current coordinate system of the specified NDF. In addition, the clump sizes and volumes are specified in WCS units. Note, the sizes are still measured parallel to the pixel axes, but are recorded in WCS units rather than pixel units. Celestial coordinate positions are units of degrees, sizes are in units are arc-seconds, and areas in square arc-seconds. Spectral coordinates are in the units displayed by the KAPPA command "ndftrace".

If the data has less than 3 pixel axes, then the columns describing the missing axes will not be present in the catalogue.

The catalogue inherits any WCS information from the NDF supplied for parameter LIKE

The "size" of the clump on an axis is the RMS deviation of each pixel centre from the clump centroid, where each pixel is weighted by the correspinding pixel data value. This excludes the instrumental smoothing specified by BEAMFWHM and VELFWHM.

The KAPPA command "listshow" can be used to draw markers at the central positions of the clumps described in a catalogue. For instance, the command "listshow fred plot=mark" will draw markers identifying the positions of the clumps described in file fred.FIT, overlaying the markers on top of the currently displayed image. Specifying "plot=STCS" instead of "plot=mark" will cause the spatial outline of the clump to be drawn if it is present in the catalogue (see parameter SHAPE).

PARDIST = LITERAL (Read)

The shape of the distribution from which clump parameter values are chosen. Can be "Normal", "Uniform" or "Poisson". The distribution for each clump parameter is specified by its own ADAM parameter containing two values; the mean and the width of the distribution. If PARDIST is "Normal", the width is the standard deviation. If PARDIST is "Uniform", the width is half the range between the maximum and minimum parameter values. In either of these two cases, if a width of zero is supplied, the relevant parameter is given a constant value equal to the specified mean. If PARDIST is "Poisson", the width is ignored. [current value]

PEAK(2) = _REAL (Read)

Defines the distribution from which the peak value (above the local background) of each clump is chosen. See parameter PARDIST for additional information. [current value]

RMS = _REAL (Read)

The RMS (Gaussian) noise to be added to the output data. [current value]

SHAPE = LITERAL (Read)

Specifies the shape that should be used to describe the spatial coverage of each clump in the output catalogue. It can be set to "None", "Polygon" or "Ellipse". If it is set to "None", the spatial shape of each clump is not recorded in the output catalogue. Otherwise, the catalogue will have an extra column named "Shape" holding an STC-S description of the spatial coverage of each clump. "STC-S" is a textual format developed by the IVOA for describing regions within a WCS - see http://www.ivoa.net/Documents/latest/STC-S.html for details. These STC-S desriptions can be displayed by the KAPPA:LISTSHOW command, or using GAIA. Since STC-S cannot describe regions within a pixel array, it is necessary to provide an NDF to define the WCS (using parameter LIKE) if using this option. An error will be reported if the WCS in the NDF does not contain a pair of celestial sky axes.

- Polygon: Each polygon will have, at most, 15 vertices. If the data is 2-dimensional, the polygon is a fit to the clump's outer boundary (the region containing all godo data values). If the data is 3-dimensional, the spatial footprint of each clump is determined by rejecting the least significant 10% of spatial pixels, where "significance" is measured by the number of spectral channels that contribute to the spatial pixel. The polygon is then a fit to the outer boundary of the remaining spatial pixels.

- Ellipse: All data values in the clump are projected onto the spatial plane and "size" of the collapsed clump at four different position angles - all separated by 45 degrees - is found (see the OUTCAT parameter for a description of clump "size"). The ellipse that generates the same sizes at the four position angles is then found and used as the clump shape.

In general, "Ellipse" will outline the brighter, inner regions of each clump, and "Polygon" will include the fainter outer regions. ["None"]

TRUNC = _REAL (Read)
The level (above the local background) at which clumps should be truncated to zero, given as a fraction of the noise level specified by RMS. [current value]

UBND() = _INTEGER (Read)
The upper pixel bounds of the output NDF. The number of values supplied (1, 2 or 3) defines the number of pixel axes in the output NDF (an error is reported if the number of values supplied for LBND and UBND differ). If an NDF is supplied for parameter LIKE, the suggested defaults for this parameter will be the upper pixel bounds of the supplied NDF.

VELFWHM = _REAL (Read)
The FWHM of the Gaussian velocity resolution of the instrument, in pixels. The generated clumps are smoothed on the velocity axis with a Gaussian beam of this FWHM, before noise is added. No velocity smoothing is performed if VELFWHM is zero. [current value]

VGRAD1(2) = _REAL (Read)
Defines the distribution from which the projection of the internal velocity gradient vector onto pixel axis 1 of each clump is chosen. Values should be supplied in dimensionless units of "velocity pixels per spatial pixel". See parameter PARDIST for additional information. [current value]

VGRAD2(2) = _REAL (Read)
Defines the distribution from which the projection of the internal velocity gradient vector onto pixel axis 2 of each clump is chosen. Values should be supplied in dimensionless units of "velocity pixels per spatial pixel". See parameter PARDIST for additional information. [current value]

Notes:

- If 3D data is created, pixel axes 1 and 2 are the spatial axes, and pixel axis 3 is the velocity axis.
- The positions of the clumps are chosen from a uniform distribution on each axis.

OUTLINECLUMP Draw an outline around a 2-dimensional clump identified by CUPID OUTLINECLUMP

Description: This procedure will outline a specified clump previously identified by CUPID:FINDCLUMPS or CUPID:EXTRACTCLUMPS. The data must be 2-dimensional, and the image over which the outline is to be drawn must have been displayed previously using KAPPA:DISPLAY.

Usage:

```
outlineclump ndf index [style]
```

Parameters:

NDF = NDF (Read)
The name of the NDF containing the clump information. This NDF should have been created using the CUPID:FINDCLUMPS or CUPID:EXTRACTCLUMPS command. The clump cut-out images contained in the CUPID extension of this NDF will be used to define the outline of the clump.

INDEX = _INTEGER (Read)
The integer index of the clump to be identified.

STYLE = LITERAL (Read)
A group of attribute settings describing the plotting style to use for the outline.

A comma-separated list of strings should be given in which each string is either an attribute setting, or the name of a text file preceded by an up-arrow character "^". Such text files should contain further comma-separated lists which will be read and interpreted in the same manner. Attribute settings are applied in the order in which they occur within the list, with later settings overriding any earlier settings given for the same attribute.

Each individual attribute setting should be of the form:

<name>=<value>

where <name> is the name of a plotting attribute, and <value> is the value to assign to the attribute. Default values will be used for any unspecified attributes. All attributes will be defaulted if a null value (!) is supplied. See section "Plotting Attributes" in SUN/95 for a description of the available attributes. Any unrecognised attributes are ignored (no error is reported).

The appearance of the clump outline is controlled by the attributes Colour(Curves), Width(Curves), etc (the synonym Contours may be used in place of Curves). The contour appearance established in this way may be modified using parameters PENS, PENROT and DASHED. [current value]

Examples:

```
outlineclump m51b 2 style="'colour=blue,width=4'"
```

This draws an outline of the second clump (as stored in m51b.more.cupid.clumps(2).model) on the current graphics device, using a blue line of four times the default thinkness.

Notes:

- The script is simply a wrapper for the KAPPA command:

contour ndf="$ndf.more.cupid.clumps($index).model" labpos=\! mode=good clear=no