# FellWalker - a Clump Identification Algorithm

David S. Berry[a,*]

[a] *Joint Astronomy Centre, 660 N. A'ohōkū Place, Hilo, HI 96720, USA*

**Abstract**

This paper describes the FellWalker algorithm, which segments a 1-, 2- or 3-dimensional array of data values into a set of disjoint clumps of emission, each containing a single significant peak. Pixels below a nominated constant data level are assumed to be background pixels and are not assigned to any clump. FellWalker is thus equivalent in purpose to the CLUMPFIND algorithm. However, unlike CLUMPFIND, which segments the array on the basis of a set of evenly-spaced contours and thus uses only a small fraction of the available data values, the FellWalker algorithm is based on a gradient-tracing scheme which uses all available data values.

*Keywords:*
clump identification, Starlink

## 1. Introduction

The CLUMPFIND algorithm (Williams et al., 1994, ascl:1107.014) has been widely used for decomposing 2- and 3-dimensional data into disjoint clumps of emission, each associated with a single significant peak. It is based upon an analysis of a set of evenly spaced contours derived from the data array and has two main parameters - the lowest contour level, below which data is ignored, and the interval between contours. However it has been noted by Pineda et al. (2009) that the decomposition produced by CLUMPFIND can be very sensitive to the specific value used for the contour interval, particularly for 3-dimensional data and crowded fields. The choice of an optimal contour interval is a compromise - real peaks may be missed if the interval is too large, but noise spikes may be interpreted as real peaks if the interval is too small.

The FellWalker algorithm attempts to circumvent these issues by avoiding the use of contours altogether. Only a small fraction of the available pixel values fall on the contour levels used by CLUMPFIND - the majority fall *between* these levels and so will have no effect on the resulting decomposition. By contrast, FellWalker makes equal use of all available pixel values above a stated threshold.

The name "Fell Walker" relates to the popular British pass-time of walking up the hills and mountains of northern England, particularly those of the Lake District (http://en.wikipedia.org/wiki/Hillwalking), and was chosen to reflect the way in which the algorithm proceeds iteratively by following an upward path from a low-valued pixel to a significant summit or peak in data-value. The following description of the algorithm uses this fell-walking metaphor at frequent intervals.

*Corresponding author
*Email address:* `d.berry@jach.hawaii.edu` (David S. Berry)

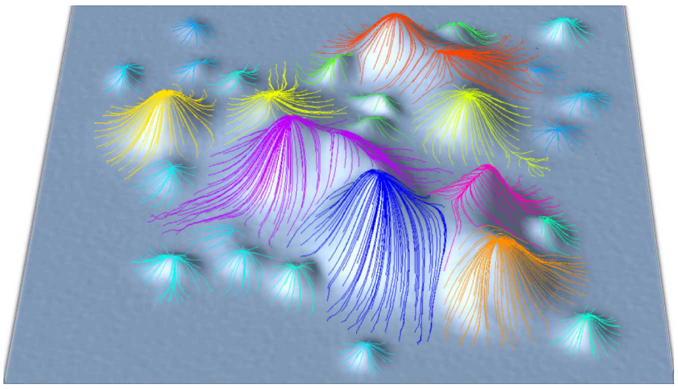

Figure 1: In 2-dimensions, peaks in data value are reminiscent of the fells of northern England. The FellWalker algorithm performs many walks starting at various low-land pixels, and for each one follows a line of steepest ascent until a significant summit is reached. All walks that terminate at the same peak are assigned to the same clump.

## 2. The FellWalker Algorithm

The core of the FellWalker algorithm consists of repeatedly following different paths of steepest ascent in order to reach a significant summit, each of which is associated with a clump, as illustrated in Fig. 1. Every pixel with a data value above a user-specified threshold is used in turn as the start of a "walk". A walk consists of a series of steps, each of which takes the algorithm from the current pixel to an immediately neighbouring pixel of higher value, until a pixel is found which is higher than any of its neighbours. When this happens, a search for a higher pixel is made over a larger neighbourhood. If such a pixel is found the walk continues from this higher pixel. If no higher pixel is found it is assumed that a new summit has been reached - a new clump identifier is issued and all pixels visited on the walk are assigned to the new clump. If at any point a walk encounters a pixel which has already been assigned to a clump, then all pixels so far visited on the walk are assigned to that same clump and the walk terminates.

It is possible for this basic algorithm to fragment up-land plateau regions into lots of small clumps which are well separated spatially but have minimal dips between them. The raw clumps identified by the above process can be merged to avoid such fragmentation, on the basis of a user-specified minimum dip between clumps. These merged clumps may, optionally, be cleaned by smoothing their boundaries using a single step of a cellular automaton.

Finally, each clump is characterised using a number of statistics, and a catalogue of clumps statistics is created together with a pixel mask identifying the clump to which each pixel is assigned.

The following sections give more detailed descriptions of each of these phases in the FellWalker algorithm.

### *2.1. Identifying Raw Clumps*

An array of integer values is first allocated, which is the same shape and size as the supplied data array. This "clump assignment array" (CAA) is used to record the integer identifier of the clump, if any, to which each pixel has been assigned. All clump identifiers are greater than zero. An initial pass is made through the supplied data array to identify pixels which have data values above a user-specified threshold value. Such pixels are assigned a value of zero in the CAA indicating that the pixel is usable but has not yet been assigned to a clump, and all other pixels are assigned a value of -1 indicating that the pixel is unusable and should never be assigned to a clump.

This initial CAA is then searched for any isolated individual pixels above the threshold. Such pixels are set to -1 in the CAA, indicating they should be ignored.

The main loop is then entered, which considers each pixel in turn as the potential start of a walk to a peak. Pixels which have a non-zero value in the CAA are skipped since they have either already been assigned to a clump (if the CAA value is positive) or have been flagged as unusable (if the CAA value is negative). A single walk consists of stepping from pixel to pixel until a pixel is reached which is already known to be part of a clump, or significant isolated peak is encountered. The vector indicies of the pixels visited along a walk are recorded in a temporary array so that they can be identified later.

At each step, the pixel values within a box of width three pixels are compared to the central pixel to find the neighbouring pixel which give the highest gradient[1]. Thus 2 neighbours are checked if the data is 1-dimensional data, 8 are checked if the data is 2-dimensional and 26 are checked if the data is 3-dimensional.

If the highest gradient found above is greater than zero - that is, if there is an upward route out of the current pixel - the walk steps to the selected neighbouring pixel. If this new pixel has already been assigned to a clump (*i.e.* if the CAA holds a positive value at the new pixel), then the new walk has joined an

[1]This gradient takes into account the fact that corner pixels are further away from the box centre than mid-side pixels.

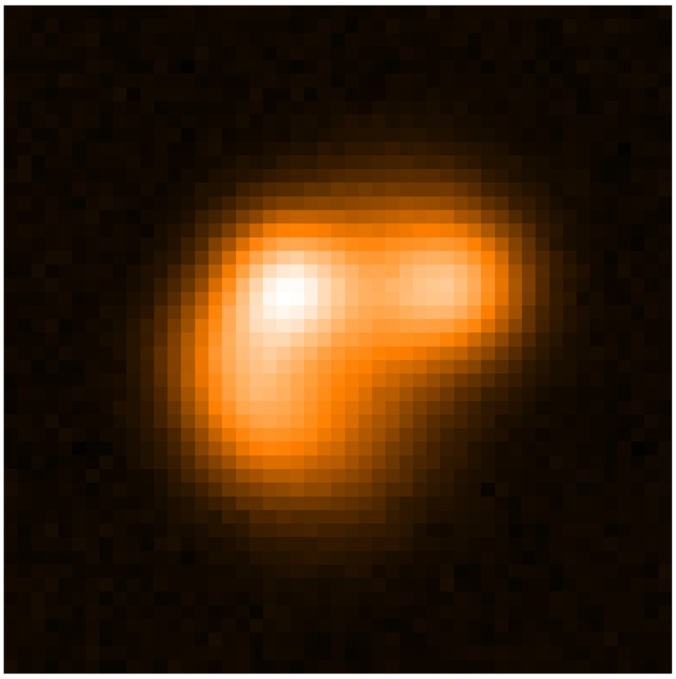

Figure 2: A 50x50 array of artificial data used to illustrate the FellWalker algorithm below.

older walk and so will eventually end up at the same peak as the older walk. The existing positive CAA value of the new pixel (*i.e.* the clump index assigned to the older walk) is copied into the CAA for all pixels visited so far on the new walk, and a new walk from the next starting pixel is initiated.

If the highest gradient found to any neighbouring pixel is less than or equal to zero, then there is no upward route from the central pixel. This could mean the walk has reached a significant peak, but it could also mean it has merely reached a noise spike. To distinguish these two cases, a search is made over a larger box[2]. If the maximum pixel value in this larger box is smaller than the central pixel value, then the central pixel is considered to be a significant peak. A new clump identifier is issued for it and stored in the CAA at all pixels visited on the walk. A new walk from the next starting pixel is then initiated.

If the maximum pixel value found in the larger box is greater than the central pixel value, then the central pixel is considered to be a noise spike. The walk then "jumps across the gap" and continues from the highest pixel found in the box.

The above process results in the CAA holding a clump identifier for every usable pixel in the supplied data array. However, some of the walks performed above may start with a section of very low gradient before any significant ascent begins. The user is allowed to specify a minimum gradient which must be achieved before a walk is considered to have begun. Any section of the walk that occurs before the first such "steep" section is flagged as unusable in the CAA. For this test, the gradient of a walk is averaged over four consecutive steps.

This process is illustrated in Fig. 3 which shows two example up-hill walks produced by FellWalker for the artificial data

[2]By default a box of width 9 pixels, but the user can specify a different size.

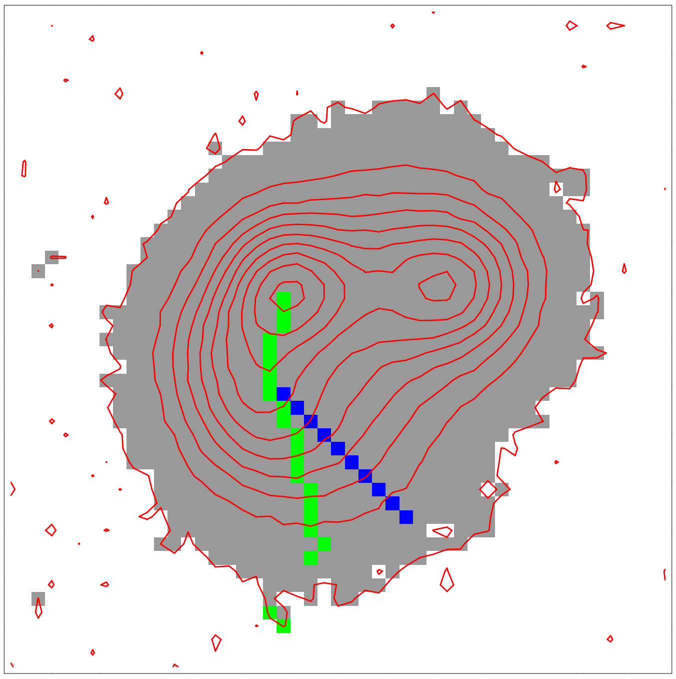
Figure 3: Two walks up a peak within the artificial data shown in Fig. 2. The contours show the data values themselves. The white pixels are below the nominated threshold, the grey pixels are above the threshold but have not yet been assigned to a clump. The green pixels trace the first walk that reached the peak. The blue pixels trace a later walk to the same peak that was terminated when it met the first walk. The green and blue pixels are all assigned to the same clump. These walks follow the steepest line of ascent. Note the gap in the green line near its start at the lowest contour - this is where a jump was made from a noise spike to the highest value in a 9x9 box of neighbouring pixels.

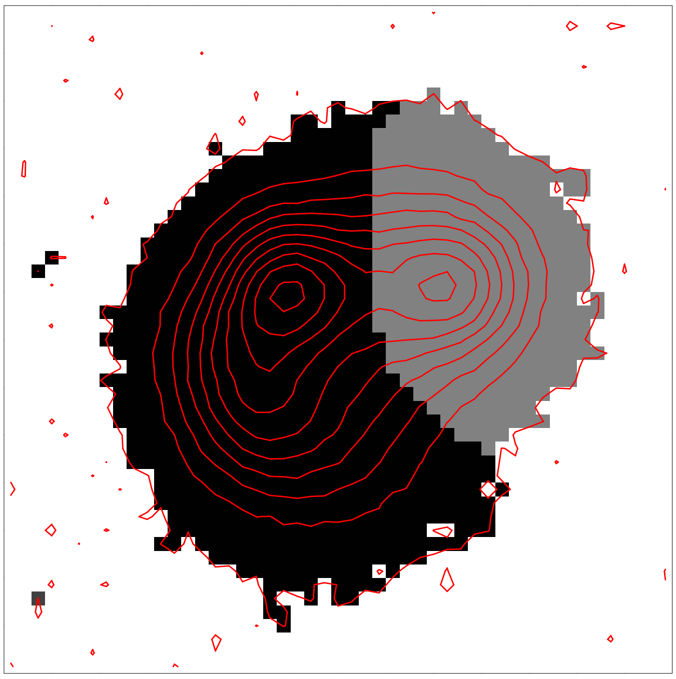
Figure 4: The raw clump mask created for the artificial data shown in Fig. 2. Two clumps are found, indicated by the black and grey pixels.

shown in Fig. 2. The final CAA produced by the above process for this data is shown in Fig. 4.

### 2.2. *Merging Clumps*

The number of significant peaks found by the above process is determined primarily by the maximum distance a walk can jump when searching for a higher neighbouring pixel value. This parameter - known as *MaxJump* - defaults to 4 pixels. Using a larger value results in more local peaks being interpreted as noise spikes, with a corresponding reduction in the number of significant peaks found. Thus peaks are discriminated simply on the basis of their spatial separation.

This means it is possible for a clump with a wide, flat summit to be fragmented into multiple clumps on the basis of any noise spikes that are separated by more than *MaxJump* pixels.

To correct this, FellWalker merges adjacent clumps if the "valley" between the two adjacent peaks is very shallow.

Each clump (the "central" clump) is considered in turn to see if it should be merged with one of its neighbouring clumps. The height of the "col[3]" between the central clump and each neighbouring clump is found in turn, and the neighbouring clump with the highest col is selected as a candidate for merging. If the peak value in the central clump is less than a specified value, *MinDip*[4], above the col, the two clumps are merged into a single clump.

Once all central clumps have been checked in this way, the whole process is repeated to see if any of the merged clumps should themselves be merged. This process repeats until no further clumps can be merged.

[3]The highest point on the boundary between the two clumps.

[4]The default is three times the noise level in the data.

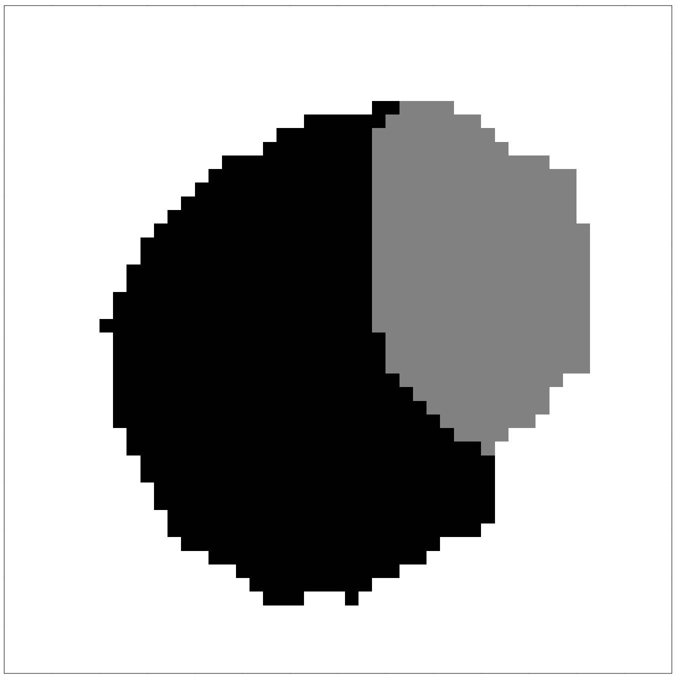

Figure 5: The smoothing effect of a single step of the cellular automaton on the clump outlines shown in Fig. 4.

### 2.3. *Cleaning Clump Outlines*

Once neighbouring clumps separated by shallow valleys have been merged, there is an option to smooth the boundaries between adjacent clumps to reduce the effects of noise. This is done using a specified number of steps of a cellular automaton to modify the integer values in the CAA.

A single step creates a new CAA from the old CAA. Each pixel in the new CAA is set to the most commonly occurring clump index within a box of width 3 pixels centred on the corresponding pixel within the old CAA. The output CAA from one step becomes the input CAA to the next step. By default, only one step is performed. Fig. 5 shows the effects of applying a single step to the CAA shown in Fig. 4.

### 2.4. *Characterising Each Clump*

## 3. CUPID - an Implementation of FellWalker

The Starlink CUPID package (Berry et al., 2007; Berry, 2013, ascl:1311.007) provides implementations of various clump-finding algorithms, including FellWalker and CLUMPFIND. In common with the rest of the Starlink software (Berry et al., 2013, ascl:1110.012), the source code for the CUPID package is open-source and is available on github at https://github.com/Starlink

## 4. Comparing FellWalker and CLUMPFIND

Include some references to earlier investigations? (e.g. Watson, 2010)

May want to consider (Westerlund et al., 2012) which uses DUCHAMP (Whiting, 2012, ascl:1201.011).

### 4.1. *Simulated Data*

### 4.2. *Real Sub-millimetre Data*

## 5. Acknowledgements

The Starlink software is currently maintained by the Joint Astronomy Centre, Hawaii with support from the UK Science and Technology Facilities Council.

## References

Berry, D.S., 2013. CUPID – A 3D Clump Identification and Analysis Package. Starlink User Note 255, Starlink Project, STFC.

Berry, D.S., Currie, M., Jenness, T., Draper, P., Bell, G., Tilanus, R., 2013. Starlink 2012: The Kapuahi Release, in: Friedel, D. (Ed.), Astronomical Data Analysis Software and Systems XXII, ASP, San Francisco. volume 475 of *ASP Conf. Ser.*. pp. 247–250.

Berry, D.S., Reinhold, K., Jenness, T., Economou, F., 2007. CUPID: A Clump Identification and Analysis Package, in: Shaw, R.A., Hill, F., Bell, D.J. (Eds.), Astronomical Data Analysis Software and Systems XVI, ASP, San Francisco. volume 376 of *ASP Conf. Ser.*. pp. 425–428.

Pineda, J.E., Rosolowsky, E.W., Goodman, A.A., 2009. The Perils of Clumpfind: the Mass Spectrum of Substructures in Molecular Clouds. Astrophys J 699, L134–L138. doi:10.1088/0004-637X/699/2/L134.

Watson, M.E., 2010. Assessing The Performance of Sub-Millimetre Compact Object Detection Algorithms. Master's thesis. University of Hertfordshire. http://www.star.herts.ac.uk/~mat/Mark_Watson_MSc_Thesis.pdf.

Westerlund, S., Harris, C., Westmeier, T., 2012. Assessing the Accuracy of Radio Astronomy Source-Finding Algorithms. Publ. Astron. Soc. Aust. 29, 301–308. doi:10.1071/AS11049, arXiv:1201.3690.

Whiting, M.T., 2012. DUCHAMP: a 3D source finder for spectral-line data. Mon Not R Astron Soc 421, 3242–3256. doi:10.1111/j.1365-2966.2012.20548.x, arXiv:1201.2710.

Williams, J.P., de Geus, E.J., Blitz, L., 1994. Determining structure in molecular clouds. Astrophys J 428, 693–712. doi:10.1086/174279.