Supporting information

Brief description on the program '*SBalgorithm*'

The actual implementation of the algorithm can be done in numerous ways, but below is a small scheme explaining the implementation in provided Matlab script.

The program to be run is *SBalgorithm.m*, where the input parameters are given in the top of the script. This program calls 4 subroutines; *data_creator.m, peak_determination.m, singlepixelcorrection.m* and *rim.m*, which will be described below. The output is an estimation of the background, the integrated intensity and the standard deviation of the integrated intensity. The theoretical model values are provided as well.

data_creator.m: Creates a Poisson-randomized data set from the input parameters given in *SBalgorithm.m*

peak_determination.m: This program comprises the actual algorithm, i.e. it eliminates the pixel with the highest value until the Poisson condition is fulfilled. A copy of the original full data set is used for all calculations and a logical matrix is used for the labeling.

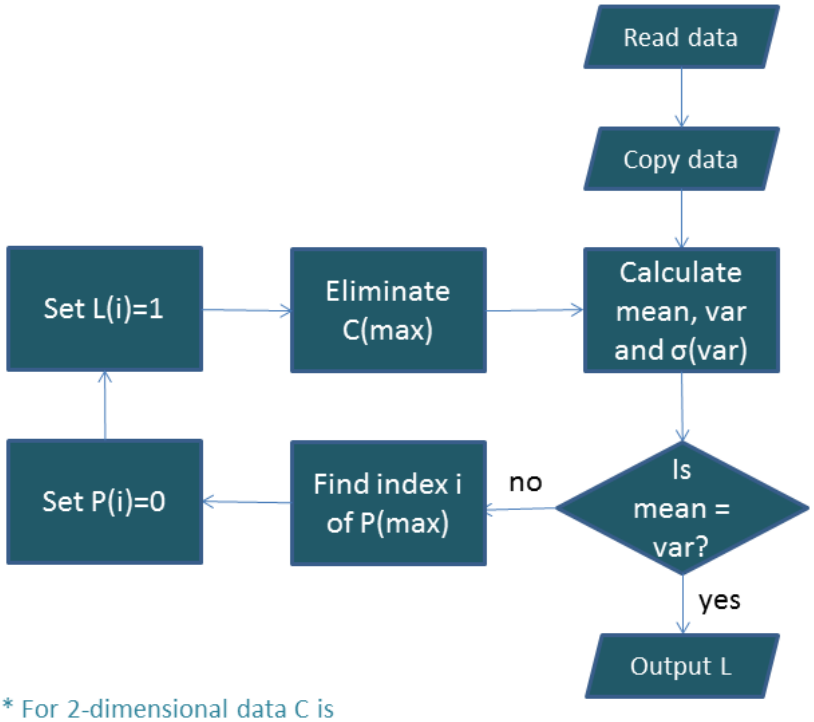
singlepixelcorrection.m: Corrects for outliers and for the 1-d case also for cavities in the peak area. If a stronger connectivity criterion than 8-fold connectivity is advisable, this subroutine should be modified.

rim.m: Adds the number of rims specified by the input parameters.

The output of each subroutine is a logical mask L_{out} of e.g. the peak region, which subsequently is multiplied by the original data set to get the relevant information. A scheme of the subroutine *peak_determination.m* is illustrated below. Before the iterative step is applied the original data of size N is copied into two arrays, P and C. A logical array L of size N is created. The variance and the mean of the data are calculated from C. If the Poisson condition is not fulfilled the elimination procedure is initiated. At iteration step i the array P is used for searching for the maximum pixel intensity (index and value) and when found this pixel value is set to zero, while the information on the index is passed on to L, where the pixel value with that index is changed from 0 to 1, indicating an s-pixel. In C this pixel index is eliminated and the variance and the mean of the remaining data are calculated. The length of P and L is thus conserved, while the length of C is equal to N-i at the i'th iteration step.

When the variance equals the mean the L array is outputted. The s-pixels are found by multiplying the data with L, whereas the b-pixels are found by multiplying the data by abs(L-1).

In the case of two-dimensional data of size $N \times M$, the arrays P and C will be of length $N*M$ at step 0, whereas L would have the size $N \times M$. One could alternatively do a search for the maximum pixel value in two dimensions.



P and C: copies of data*
L: logical array* with size(data)

Within k standard deviations of the variance

L*data gives the s-pixels, abs(L-1)*data gives the b-pixels

* For 2-dimensional data C is converted to a vector.