ABLE documentation

In this document, we provide guidance on how to implement ABLE. For further technical details, see Reynolds et al. (2016), ABLE: an activity-based level set segmentation algorithm for two-photon calcium imaging data.

The process is as follows:

1. **INITIALISATION:** The segmentation algorithm is initialised using the function '*initialise.m*'. This function identifies peaks in the correlation image (and/or mean image) as candidate ROIs. This algorithm has one important tuning parameter that defines how conservative the initialisation is (i.e. how high a peak must be with respect to neighbouring pixels for it to be selected.) The higher the value of alpha, the more conservative the initialisation. This parameter can be tuned with the function '*tune_alpha.m*'.

2. **SEGMENTATION:** The segmentation is performed using the function '*segment.m'.* In the segmentation algorithm, each ROI identified in the initialisation is represented by a separate Level Set function (LSF). This is a function that is positive for all pixels inside an ROI, negative for pixels outside and zero for pixels on the boundary. ABLE iteratively updates the LSF in a manner that minimises a cost function. The cost function includes two terms: a data-based term (which aims to assign a pixel to the cell interior or exterior depending on which subregion its temporal activity is most similar to) and a regularization term (which maintains a smooth level set function). There is one important parameter to set for the segmentation algorithm, lambda – this defines the relative weight of the data-based term and regularization term. The higher the value of lambda, the higher the relative weight of the data-based term. This parameter can be tuned with the function '*tune_lambda.m*'.

In the following, we provide further detail on the four functions necessary for using ABLE: '*initialise.m*', '*tune_alpha.m*', '*segment.m'* and '*tune_lambda.m*'. The usage of these functions can be seen in the demo: '*demo.m'.*

Guidance for function *initialise.m*

SUMMARY: This function identifies peaks in a summary image (usually the correlation image) as candidate ROIs. This algorithm has one important tuning parameter that defines how conservative the initialisation is (i.e. how high a peak must be with respect to neighbouring pixels for it to be selected.) Optionally, a user can also specify a secondary summary image (usually the mean image). In that case, peaks in either image are identified as candidate ROIs.

INPUTS: The following inputs are required:

Name	Data type	Explanation
Metric	M x N array	A summary image, usually the correlation image. The peaks in this image are identified by the algorithm as candidate ROIs.
Radius		The expected radius of a cell.
Alpha		The value of the tuning parameter, which defines the relative height of peaks that are suppressed. Usually in the range [0.1,1].
Options	struct	Initialisation options. If you prefer not to specify any options then input this as an empty vector: [].

OPTIONAL INPUTS: The input variable 'options' is a structure that specifies options of the initialisation algorithm. None of the fields are compulsory. The following table details the possible fields of this variable:

Field name	Explanation
blur_radius	Default value is 1. This is the radius (in pixels) of the blurring applied to the input summary image ('metric') and the secondary summary image, if this is provided.
secondary_metric	This is an MxN array, which corresponds to a summary image, e.g. the mean image. If this field exists, initialisation is performed on both 'metric' and the second summary image. Peaks in either image are selected as candidate ROIs.
second_alpha	Compulsory if the field secondary_metric exists. This is the value of alpha (see first table) used for the second summary image.

OUTPUT: The output ('masks') is an M x N x K array, where K is the number of ROIs found. The i^{th} sheet (masks(:,:,i)) corresponds to the i^{th} ROI. If the value at a pixel is - 1 in the i^{th} sheet, then that pixel is inside the i^{th} ROI. If it is +1, that pixel is outside the i^{th} ROI.

TIPS:

1. The function '*tune_alpha.m*' should be used to tune the value of alpha separately for both the primary summary image ('metric') and, if included, the secondary summary image ('options.secondary_metric').
2. It is usually beneficial to initialise on both the correlation image and the mean image.
3. When larger ROIs are sought (e.g. cell bodies), a blur radius of at least 1 is advisable. When the user is also interested in detecting smaller ROIs (e.g. cross-sections of dendrites, see the demo) a blur radius of 0.5 may be preferable.

Guidance for function *tune_alpha.m*

SUMMARY: This function can be used to tune the value of alpha – the parameter that defines the level of conservatism of the initialisation algorithm. The higher the value of alpha, the more conservative the initialisation. The function proceeds as follows: the initialisation for the first value of alpha is plotted, the user then has the option to input a new value of alpha. This process continues until the new value of alpha input by the user is the same as the current value, indicating that the user is sufficiently satisfied.

INPUTS: The following inputs are required:

Name	Data type	Explanation
metric	M x N array	A summary image. The peaks in this image are identified by the algorithm as candidate ROIs.
radius		The expected radius of a cell.
alpha		An initial estimate for alpha. Recommended value is 0.5.
options	struct	Initialisation options, see documentation for '*initialise.m*'. Do not include a secondary metric (i.e. the field options.secondary_metric) for tuning alpha. Tune alpha for each metric separately.

OUTPUTS: See documentation for '*initialise.m*'.

TIPS:

1. When the initialisation is performed on two summary images (i.e. options.secondary_image contains a second summary image), the value of alpha should be tuned for each image separately.

Guidance for function *segment.m*

SUMMARY: This function contains the segmentation algorithm. In short, for every ROI identified in the initialisation, the boundary estimate is iteratively updated in a manner than minimises a cost function. This cost function includes one data-driven term (which aims to assign a pixel to the cell interior or exterior depending on which subregion its temporal activity is most similar to) and a regularization term (which maintains a smooth level set function.

INPUTS: The following inputs are required:

Name	Data type	Explanation
phi_0	M x N x K array	This is the initialisation (generated by function *initialise.m*). Here, K is the number of ROIs found in the initialisation. The i^{th} sheet (phi_0 (:,:,i)) corresponds to the i^{th} ROI. If the value at a pixel is - 1 in the i^{th} sheet, then that pixel is inside the i^{th} ROI. If it is +1, that pixel is outside the i^{th} ROI.
video	M x N x T array	This is the imaging video to be segmented. It is most memory efficient if the video is not converted to double precision but rather kept in unsigned integer precision (e.g. uint8, uint16), which is usually the raw data format.
radius		The expected radius of a cell.
options	struct	Segmentation options. If you prefer not to specify any options then input as an empty vector: [].

OPTIONAL INPUTS: The input variable 'options' is a structure that specifies options of the segmentation algorithm. None of the fields are compulsory. The following table details the possible fields this variable:

Field name	Default	Explanation
lambda		This parameter defines the relative weight of the data-based term and regularizer in the cost function. The higher the value of lambda the more weight the data-based term has. This is the most important parameter and should be tuned with the function '*tune_lambda.m'*.
metric	'corr'	This is the type of metric used to compare similarity of time courses, either 'corr' for correlation or 'euclid' for the Euclidean distance.
maxIt	150	The number of iterations of the algorithm.
mergeCorr	0.8	This is correlation coefficient threshold above which two neighbouring ROIs will be merged.
mergeTime		If mergeTime = 'during', the ROIs can be merged during the update phase of the algorithm. This is the default option when the correlation metric is used. Otherwise, if mergeTime = 'atEnd', ROIs are merged at the end of the update process. This is the default option for the Euclidean distance metric.
snr	[]	This is the signal-to-noise ratio (SNR) of a cell (dB). If this field is present instead of 'mergeCorr', the correlation threshold above which ROIs are merged is inferred from the SNR.
plot_progress	[]	If this field is present, whilst the segmentation occurs, plots of progress are shown to the user. In this case a user also must

		enter the correlation and mean images as arguments to the fields options.corrIm and options.meanIm, respectively.
cell_to_monitor	K/2	If the field 'plot_progress' is present, this is the ID of the cell whose progress is plotted.

OUTPUTS: The following variables are output from the algorithm.

Field name	Data type	Explanation
cellMasks	M x N x K' array	This variable contains the segmented masks, K' is the number of ROIs found. Each sheet (cellMasks(:,:,i)) corresponds to one ROI. If the value at a pixel is 1 in the i^{th} sheet, then that pixel is inside the i^{th} ROI. If it is 0, that pixel is outside the i^{th} ROI.
cellTimeSeries	K' x T array	This variable contains the time courses of all ROIs. The i^{th} row corresponds to the average temporal activity of all pixels exclusively within the i^{th} ROI.
nhbdTimeSeries	K' x T array	This variable contains the time courses of the neighbourhoods of all ROIs. The i^{th} row corresponds to the average temporal activity of all pixels in the neighbourhood of the i^{th} ROI, excluding those in neighbouring cells.

TIPS:

1. The metric used to compare similarity of time courses is either the correlation coefficient (options.metric = 'corr') or the Euclidean distance (options.metric = 'euclid'). Using the correlation is the default and is recommended for data in which cells have varying intra-cellular and inter-cellular brightness. Datasets in which cells are relatively sparse and have high baseline fluorescence may benefit from using the Euclidean distance metric.
2. The runtime of the algorithm is virtually unaffected by the number of frames in a video – any video that can be stored in MATLAB's memory can be processed. There is no need to convert a video to double precision, longer videos can be processed if they are stored in their original unsigned integer precision (e.g. uint8, uint16), as this is more memory efficient.
3. The runtime of the algorithm (in its current implementation) is linear in the number of cells. On a standard laptop it takes approximately 15 minutes to segment 400 cells.
4. The option 'plot_progress' is good as a diagnostic tool to check whether the settings of the algorithm are appropriate. This does, however, slow down the algorithm's operation, so it is best to use this with a limited selection of neighbouring ROIs (i.e. only input a subsection of the sheets of phi_0).

Guidance for function *tune_lambda.m*

SUMMARY: This function can be used to tune the value of lambda that is used for the segmentation algorithm (‘*segment.m’).* The function proceeds as follows: for a selected active contour, the updates of the contour are plotted whilst the algorithm estimates the optimal cell boundary. Based on this, the user can then select to decrease or increase lambda. This is repeated until the user indicates that they are satisfied with the value of lambda.

INPUTS: The following inputs are required:

Name	Data type (if important)	Explanation
phi_0	M x N x K array	The initialisation (generated by function *initialise.m*). K is the number of ROIs found in the initialisation. Each sheet (masks(:,:,i)) corresponds to one ROI. If the value at a pixel is - 1 in the i^{th} sheet, then that pixel is inside the i^{th} ROI. If it is +1, that pixel is outside the i^{th} ROI.
video	M x N x T array	This is the imaging video to be segmented. Although this can be a variable of type uint or double, it is more memory efficient as uint.
radius		The expected radius of a cell.
options	struct	Segmentation options. If you prefer not to specify any options then input as an empty vector: []. See documentation for ‘*segment.m’* for further detail.
corrIm	M x N array	The correlation image.
meanIm	M x N array	The mean image.

OUTPUT: The tuned value of lambda.

TIPS:

1. The larger the value of lambda, the more weight the data-based term has with respect to the regularizer. If the contour is not evolving (the activity in the bottom row of the plot is consistently 0), lambda will need to be increased. If the contour is evolving in an unstable way (the level set function in the left-most subplot of the second row is jagged), lambda should be decreased.
2. For an example of stable evolution, see the video called ‘evolution_video.avi’.
3. For the correlation metric and an unsigned integer video, lambda is typically in the range [5,50].