AQUA DOCUMENTATION

Version Dec.2018

TABLE OF CONTENTS

1 OUTPUT FILES

After a brief intro to the output files, details are given to the main output files and features.

1.1 OVERVIEW OF OUTPUT

In this section, we briefly overview the files exported from AQuA. All output files from AQuA are listed below.

File name	Description
`name_postfix`.mat	Main output file with features, curves, options and more
`name_postfix`.tif	Movie with events overlaid on it
`name_postfix`.xlsx	Features for events after proof reading. See below for details
`name_postfix`_landmark.png	Map for regions and landmarks
`name_postfix`_region_`i` .xlsx	features for events in region `i`. Each region will have one file

By default, `name` is the input data name and `postfix` is `AQuA`.

1.1.1 MAIN OUTPUT

The main output file in AQUA is `name_postfix.mat`. It contains a MATLAB structure called `res`. The main contents of it are shown below. You can use this structure for further analysis.

Name	Type	Size	Description
opts	struct	1	Detection parameters used in this run
ftsFilter	struct	1	Extracted features for events after proofreading
evtFilter	cell	Event number (filtered)	Events after proof reading
dffMatFilter	single	Event number (filtered) by Frames by 2	dF/F0 curves for events after proof-reading
dMatFilter	single	Event number (filtered) by Frames by 2	Raw curves for events after proof-reading
riseLstFilter	cell	Event number (filtered)	Onset time map for events after proofreading
evtSelectedList	double	Event number (filtered)	Index of proof read events w.r.t the complete list of events

The curves of all events are saved in `res.dffMat` and `res.dMat`. The former contains dF/F0 curves and the latter saves original curves (which is the average of all pixels in this event).

The features of all events are saved in a MATLAB structure `res.fts`. Each category of features is saved in a field of `res.fts`. Unless otherwise noted, the units are based on seconds and µm.

If you are only interested in filtered and selected events, use `res.dffMatFilter`, `res.dffFilter`, `res.ftsFilter` instead. The list of selected features are in `res.evtSelectedList`.

The detection parameters are saved in `res.opts`. For example, `res.opts.frameRate` and `res.opts.spatialRes` saves the spatial and temporal resolution.

1.1.2 EXPORTED FEATURES

We list all exported features and show how they are calculated from basic features in `res.fts` or `res.ftsFilter`. We also briefly describe them. More details for basic features can be found in the features section. These features are used in the overlay function of AQuA. While in the overlay function, propagation direction of interest can be selected in the drop-down menu, here all four directions are listed. Some features do not have units.

Name	**Unit**	**Description**
Index		Event index
Basic - Area	µm^2	Area in µm^2
Basic - Perimeter	µm	Perimeter in µm
Basic - Circularity		Closeness to a circle, 1 is the closest, the larger, the less circular
Curve - P Value on max Dff (-log10)		Compare dF/F0, baseline and noise level to find significance level
Curve - Max Dff		Peak of the dF/F0 curve for current event
Curve - Duration 50% to 50%	seconds	50% onset time point to 50% offset time point, in seconds
Curve - Duration 10% to 10%	seconds	10% onset point to 10% offset point, in seconds. Less robust than 50% to 50%
Curve - Rising duration 10% to 90%	seconds	Onset duration from 10% to 90%, in seconds
Curve - Decaying duration 90% to 10%	seconds	Offset duration from 90% to 10%
Curve - Decay tau	seconds	Decay time by fitting exponential curve on df/f0
Propagation - onset - overall	µm	Single event propagation onset score, sum of four

		directions
Propagation - onset - one direction - Anterior	µm	Single event propagation onset distances in anterior direction
Propagation - onset - one direction - Posterior	µm	
Propagation - onset - one direction - Left	µm	
Propagation - onset - one direction - Right	µm	
Propagation - onset - one direction - ratio - Anterior	µm	Single event propagation onset distances in anterior direction, normalized by the sum of four directions
Propagation - onset - one direction - ratio - Posterior	µm	
Propagation - onset - one direction - ratio - Left	µm	
Propagation - onset - one direction - ratio - Right	µm	
Propagation - offset - overall	µm	Single event propagation offset score, sum of four directions
Propagation - offset - one direction - Anterior	µm	Single event propagation offset distances in anterior direction
Propagation - offset - one direction - Posterior	µm	
Propagation - offset - one direction - Left	µm	
Propagation - offset - one direction - Right	µm	
Propagation - offset - one direction - ratio - Anterior	µm	Single event propagation offset distances in anterior direction, normalized by the sum of four directions
Propagation - offset - one direction - ratio - Posterior	µm	
Propagation - offset - one direction - ratio - Left	µm	
Propagation - offset - one direction - ratio - Right	µm	
Landmark - event average distance - landmark 1	µm	Average distance of an event centroid to a landmark across all frames

Landmark - event minimum distance - landmark 1	µm	Minimum distance of an event centroid to a landmark across all frames
Landmark - event toward landmark - landmark 1	µm^3	Score for propagating toward a landmark
Landmark - event away from landmark - landmark 1	µm^3	Score for propagating away from a landmark
Landmark - event toward landmark before reaching - landmark 1	µm^3	Score for propagating toward a landmark before it reaches the landmark
Landmark - event away from landmark after reaching - landmark 1	µm^3	Score for propagating away from a landmark after it reaches the landmark
Region - event centroid distance to border	µm	Distance from event centroid to the boundary of the region
Region - event centroid distance to border - normalized by region radius		Divide the distance to region boundary by the radius of the region
Network - Temporal density		Number of events that share spatial footprint with current event (including itself)
Network - Temporal density with similar size only		Number of events that share spatial footprint with current event and have similar size (50% to 200%)
Network - Spatial density		Number of events co-occurred with current event

In either `name_postfix`.xlsx or `name_postfix`_region_`i` .xlsx, the same set of features are reported.

1.2 OUTPUT DATA STRUCTURE

The complete list of fields for `res` in main output file is shown below. Some items are further detailed in the next sub-section. More details for features can be found in the features section.

Proof reading includes:

- Removed events outside the given range of selected features
- Removed events outside user-drawn regions
- Manual removal of events

Name	Type	Size	Description
opts	struct	1	Detection parameters
scl	struct	1	GUI struct for movie brightness contrast and more

btSt	struct	1	GUI struct for component status
ov	containers.Map	8	GUI struct for overlays in different steps
bd	containers.Map	3	GUI struct for regions, landmarks and other boundaries
datOrg	uint16	Height by Height by Frames	Raw data after square root
evt	cell	Event_number	Detected events linear location
fts	struct	1	Extracted features
dffMat	single	Event_number by Frames by 2	dF/F0 curves
dMat	single	Event_number by Frames by 2	Raw curves
riseLst	cell	Event_number	Onset time maps
featureTable	table	Feature numbers	Selected features for export
userFeatures	table	Feature (combined) numbers by 2	Selected features for overlay function
ftsFilter	struct	1	Extracted features for events after proof reading
evtFilter	cell	Event number (filtered)	Events after proof reading
dffMatFilter	single	Event number (filtered) by Frames by 2	dF/F0 curves for events after proof reading
dMatFilter	single	Event number (filtered) by Frames by 2	Raw curves for events after proof reading
riseLstFilter	cell	Event number (filtered)	Onset time map for events after proof reading
evtSelectedList	double	Event number (filtered)	Index of proofread events w.r.t the complete list of events
maxVal	single	1	Movie scale
stg	struct	1	AQuA pipeline stages

1.2.1 FIELDS IN **RES** STRUCTURE

opts

Parameters for event detection and data itself. More details can be found in parameter sections.

ov

A MATLAB container map for overlays from different stages. Container map is similar to a dictionary in other programming languages. These overlays are mainly used for the overlay functions in AQuA. This helps to visualize the results from each step of the detection pipeline. We define an *overlay component* as a spatial temporal component, like super voxels, super events and events.

`None`

No overlay.

`Step 1: active voxels`

Foreground voxels overlay.

`Step 2: super voxels`

Super voxels overlay.

`Step 3a: super events`

Super events overlay.

`Step 3b: events all`

Event overlay. Generated with 3a in event detection.

`Step 4: events cleaned`

Event overlay after cleaning based on z-scores

`Step 5: events merged`

Events overlay after merging.

`Events`

Final events overlay.

ov (each key)

Each item (or key) in the `ov` map is a structure containing the following fields:

`frame`

Overlay information for each frame. For T frames, this is a T by 1 cell array. Each element contains all overlays at *one frame*. Assume we have M components in this frame.

`idx`: M by 1 vector. The event index for each component.

`pix`: M by 1 cell array. Pixels (2D linear index) for each component

`val`: M by 1 cell array. Overlay intensity for each pixel for each component

`idx`

All indices of overlay components, like super voxels and events.

`idxValid`

Non-empty indices of overlay components, like super voxels and events.

`col`

Color for each overlay component. For N events, this is a N by 3 matrix. Each row is the RBG triplet. Each element ranges from 0 to 1.

`colVal`

Value of the color if `colorCodeType` is set to `Linear`.

`sel`

Selected events. For N events, this is a N by 1 vector. If the value for a component is set to 0, that component will not be shown.

`name`

Name for current overlay. Like 'Events'.

`colorCodeType`

Color coding scheme, which can be set to `Random` or `Linear`.

Use random to use random colors to represent different events.

Use linear to use the shade of the color to visualize feature values.

bd

It is a map containing the following keys:

`cell`

The boundary of user-drawn regions, or the regions generated by mask builder

`landmk`

The boundary of user-drawn landmarks or those from mask builder.

Scl

AquA movie and overlay adjustment variables.

btSt

Status variable for AquA GUI.

Evt

Linear index for each event. This is the same as `res.fts.loc.x3D`. See `res.fts.loc.x3D` for more details.

Fts

Extracted features. See parameter sections.

dffMat

dF/F0 curves for each event. Optionally, baseline trend is corrected.

`dffMat(:,:,1)` is the dff calculated from raw data. For example, `dffMat(I,:,1)` is the curve for event `i`.

`dffMat(:,:,2)` is the dff after removing the contributions from other events The other events are removed and imputed with nearby values.

dMat

Raw curves for each event. Optionally, baseline trend is corrected.

`dMat(:,:,1)` is the raw curve calculated from raw data. For example, `dMat(i,:,1)` is the curve for event `i`.

`dMat(:,:,2)` is the raw curve after removing contributions from other events. The other events are removed and imputed with nearby values.

riseLst

Onset time map for each event. Each element for one event. It contains the following fields:

`dlyMap`

Delay map for the 2D event. The value corresponds to the time step. A fraction of a frame is possible. We use a patch taken from the whole field of view. The range of the patch is given by `rgh` and `rgw`.

`rgh`

The height range of the event patch relative to the raw movie.

`rgw`

The width range of the event patch relative to the raw movie.

stg

A field with two sub fields.

`stg.detect`

Whether detection has finished or not. 0: not finished. 1: finished.

`stg.post`

Whether the whole pipeline has finished or not. 0: not finished. 1: finished.

ftsFilter, evtFilter, dffMatFilter, dMatFilter, riseLstFilter

The same as their counterparts without 'Filter', except they only contain events after proofreading.

featureTable, userFeatures

See feature sections.

1.3 FEATURES

All features are stored in `fts` or `ftsFilter` fields of the res structure in output MAT file This section describes the structure of this field and explains the features.

We assume we are using `res.fts`. All items listed below are inside it. For example, when we talk about `loc.x3D`, we either refer to `res.fts.loc.x3D` or `res.ftsFilter.loc.x3D`.

You can find the same set of items in `res.ftsFilter`. Some selected fields will be described in more details.

Most features use units `um` and `seconds`. If some features are used to indicate the locations in the movie, we use units like `pixel`, `voxels` and `frames`.

`N`

Events number

`L`

Landmark number

`R`

Region number

`Thr`

Threshold levels

`H`

Height of the field of view, in pixels

`W`

Width of the field of view, in pixels

`T`

Duration of the movie, in frames

1.3.1 POSITIONS, SIZE, AND SHAPE

Features in this section are related to the size, shape and locations of features.

Field	**Type**	**Size**	**Units**	**Description**
basic.map	cell	N		Cropped spatial footprint
basic.peri	double	N	µm	Perimeter of spatial footprint of events
basic.area	double	N	µm^2	Area of spatial footprint
basic.circMetric	double	N		Circularity score of spatial footprints
loc.t0	double	N	frame	First frame in an event that is bright enough (20% peak)
loc.t1	double	N	frame	Last frame in an event that is bright enough (20% peak)
loc.x3D	cell	N	pixel	Spatial temporal location in linear index
loc.x2D	cell	N	voxel	Spatial footprint in linear index
bds	cell	N	pixel	Boundary pixels for each event. Use linear index.

`basic.map`

The spatial footprint of each event. Each event is a 2D binary image, where the bright parts are events. To save space, we crop the event out as patches.

The purpose of this feature is mainly to help calculate the area and shape-based features. It may also be used for visualization.

The range of cropping is given by:

```
rgH = max(min(ih)-1,1):min(max(ih)+1,H);  % range on Y direction
rgW = max(min(iw)-1,1):min(max(iw)+1,W);  % range on X direction
```

H is the Height, W is the width. `ih` is the Y direction coordinates and `iw` is the X direction coordinates of pixels.

`basic.peri`

Perimeter of the spatial footprint of the event. The value is calculated from `basic.map` using `regionprops` function of MATLAB.

`basic.area`

The area of the spatial footprint. It is also calculated from `basic.map` using `regionprops` function of MATLAB.

`basic.circMetric`

A score for the circularity of the spatial footprint. It is calculated by C^2/4S, where C is the perimeter and S the area, as calculated above. For shapes that is closer to a circle, the value will be closer to 1. For example, for circles, the score is 1; for lines, the score can be very large.

`loc.x3D`

Exact location of all voxels. Formerly we name it locAbs. If the movie is described as a 3D volume, then each event can be described by a list of (x,y,t) coordinates. Each element in loc.x3D/locAbs contains all coordinates for an event and these coordinates are stored as a linear index (see, for example, **here**).

`loc.x2D`

A list of pixels for the 2D spatial footprint of each event. The values are given in linear index as well.

`loc.t0`, `loc.t1`

Starting and stopping frames of each event

`bds`

Each item is a cell. The cell could contain one or multiple arrays. Each array describes one outer or inner boundary of the spatial footprint (**`loc.x2D`**) of the event. The values are obtained by MATLAB function **`bwboundaries`**. For example, assume one component is a square, then we have a 4 by 2 array, and each row is the Y and X coordinates of the four vertices.

1.3.2 TIME AND CURVE RELATED

Curve related features, like duration and dF/F0 are in the field **`res.fts.curve`**. These features are based on the extracted dF/F0 curves. To compensate for the bleaching, remaining motion or other effects, the trend of the curves is also corrected. Details on these steps are described in **Details in Curve and Feature Extraction**.

Field	**Type**	**Size**	**Units**	**Description**
curve.rgt1	double	N by 2	frame	Extended start and end frames of each event.
curve.dffMax	double	N		Maximum dF/F0 for each event using raw movie
curve.dffMax2	double	N		Maximum dF/F0 for each event by excluding other events
curve.dffMaxFrame	double	N	second	The time when maximum dF/F0 occurs
curve.dffMaxZ	double	N		Z score for the maximum dF/F0
curve.dffMaxPval	double	N		Significance of maximum dF/F0 using *p*-value
curve.tBegin	double	N	frame	Same as **`loc.t0`**
curve.tEnd	double	N	frame	Same as **`loc.t1`**

curve.rise19	double	N	second	Onset time from 10% to 90% of peak intensity in dF/F0 curve
curve.fall91	double	N	second	Offset time from 90% to 10% of peak intensity in dF/F0 curve
curve.width55	double	N	second	Duration time from 50% onset to 50% offset w.r.t. peak intensity in dF/F0 curve
curve.width11	double	N	second	Duration time from 10% onset to 10% offset w.r.t. peak intensity in dF/F0 curve
curve.decayTau	double	N	second	Decay time constant obtained by fitting exponential model

`rgt1`

This is the by-product when finding a longer curve for curve-related features. We only use part the dF/F0 curve to extract curve-related features. This operation is intended to make the estimation of the curve-related features more local. This feature serves as an upper bound of duration so that we will not under-estimate the event duration.

Algorithm

Find all events that share spatial footprints with current event and occur earlier than current event. Find the one that is temporally closest to current one. Find time point **`t0`**, which is the frame with lowest value of dF/F0 between these two events. Similarly, we can find **`t1`** for events occurring later than current one. Then **`rgt1`** is **`t0:t1`**. If we cannot find either **`t0`** or **`t1`**, we use **`loc.t0`** or **`loc.t1`**.

`dffMaxZ`, `dffMaxPval`

These two scores describe the significance of the maximum dF/F0. It can also be used as a significance measure of the whole event. The larger the Z-score, the smaller the p-value, the more significant.

The Z score is calculated from the dF/F0 curve. Let F_max be the maximum value of dF/F0 in one event, let b_pre be the baseline before F_max and b_post be the baseline after F_max, then we define Z score as

Z = min(F_max-b_pre,F_max-b_post)/sigma/2,

where sigma is the estimated noise standard deviation on the dF/F0 curve for current event.

The p-value is directly calculated from the z score:

```
dffMaxPval = 1-normcdf(dffMaxZ)
```

Though usually 0.05 is set as the significance threshold, for movies with imperfect imaging conditions (in terms of noise distribution), which is always the case, the significance level should be much lower.

`rise19`, `fall91`, `width55`, `width11`

We search within the dF/F0 curve inside``rgt1`` to find these four durations. We find six time points: 10% onset time, 50% onset time, 90% onset time, 10% offset time, 50% offset time and 90% offset time.

Example

To find 50% offset time, we find all time points that are earlier than the peak and lower than or equal to 50% of the peak intensity. Then we get the one that is closest to the peak in time.

Caution

The estimates could be inaccurate for multiple reasons. First, if the dF/F0 curves are noisy, or contains some residual trend, the 10%, 50% and 90% intensity are not correctly estimated. The 10% values are especially sensitive. Second, note that each curve is the average of the event, which does not consider propagation. The time points estimated may not be the one of interest in some scenarios. Third, some peaks have very short duration, which requires interpolation to get good onset/offset time estimates. We do not have this function yet.

decayTau

The decay time constant tau is obtained by using the MATLAB `fit` function with an `exp1` model. The option for this function is:

```
foptions = fitoptions('exp1');
foptions.MaxIter = 100;
```

The curve used for fitting is transformed to make its intensity between 0 and 1.

1.3.3 SINGLE EVENT PROPAGATION

We extract this feature for individual event propagation. To characterize propagation, we binarize the movie (at each event) with multiple thresholds. The intensity thresholds used here are `minShow1:0.1:0.8`, where `minShow1` is a user-specified parameter.

The distances reported in these features are calculated based on the change of centroid locations of events at different directions along time. We use the initial centroid of the event as the starting point.

Example: distance calculation

The movie has a field of view with size 100 by 100. With a given threshold, the event has initially appeared at t=1. We find the initial centroid at coordinates (50,50). At t=2, the event grows compared to t=1, and we want to quantify the distance of propagation in the anterior direction from t=1 to t=2. Assume the anterior direction is north. Then we only look at the event in the Y range 50 to 100 and X range 1 to 100. We find the centroid of the event that is within that region. If at t=2, the centroid is at coordinates (60,55), then the anterior direction propagation distance from t=1 to t=2 is 55-50=5 pixels. We transform the units to μm when we report the features.

Definition: growing and shrinking

These features record the growing and shrinking behavior of propagation. For example, under a given intensity threshold, assume an event does not arrive at that location at time t (in other words, the intensity at that location is lower than the threshold). However, later it arrives at that location, say at time t+1 (the intensity becomes high enough), we say the event grows to that location. Even later for example, at time t+2, the signal intensity is lower than the threshold again at that location. Then we say the event shrinks. Therefore, an event can have both a growing and shrinking score at each time at a given intensity threshold.

Field	**Type**	**Size**	**Units**	**Description**
propagation.propGrow	cell	**N**	μm	Onset propagation distance per direction per frame
propagation.propGrowOverall	double	**N** by 4	μm	Onset propagation distance per direction
propagation.propShrink	cell	**N**	μm	Offset propagation distance per direction per frame
propagation.propShrinkOverall	double	**N** by 4	μm	Offset propagation distance per direction
propagation.areaChange	cell	**N**	μm^2	Event area change per frame per threshold
propagation.areaChangeRate	cell	**N**		Event area change ratio (spatial footprint as reference)
propagation.areaFrame	cell	**N**	μm^2	Event area per frame per threshold
notes.propDirectionOrder	cell	4		Definition of four propagation directions

`propagation.propGrow`

Each item in the cell is the propagation for one event. Assume the event has T0 frames, the item is a T0 by 4 matrix. The time range recorded is the same as **`loc.t0`** and **`loc.t1`**, or **`curve.tBegin`** and **`curve.tEnd`**. This feature records at each frame, in each of the four direction, the distance change compared to the previous frame. As the distance is calculated at multiple thresholds, this feature reports the largest one at each frame for each direction.

Note

We only record positive values in this feature as we are recording the growing pattern. Negative changes are interpreted as shrinking and are reported in those shrinking features.

`propagation.propGrowOverall`

Take the summation of all frames in the **`propagation.propGrow`** feature. This gives the overall growing propagation for each event in four directions.

`propagation.propShrink`, `propagation.propShrinkOverall`

Similar to **`propagation.propGrow`** and **`propagation.propGrowOverall`**, except that we record the shrinking distances.

`propagation.areaFrame`

Each item in the cell array is for one event. Assume the event has T0 frames, the item is a T0 by **`Thr`** matrix. It records the area of the event at one frame under a certain threshold.

`propagation.areaChange`

In **`propagation.areaFrame`**, by subtracting the value of previous frame from current frame, we obtain the event area change at each frame for each threshold.

`propagation.areaChangeRate`

This is a normalized version of **`propagation.areaChange`**. We divide each item in **`propagation.areaChange`** with the area of the spatial footprint. The spatial footprint contains all spatial locations the event ever visited. The area is given by **`basic.area`**. We take the **maximum for all thresholds** and report the **largest rate of change at each frame** in this feature.

`notes.propDirectionOrder`

Four directions in: anterior, posterior, lateral (left w.r.t. anterior-defined up) and medial (right w.r.t. anterior-defined up). The user can draw the anterior direction. By default, the anterior direction is north (up).

1.3.4 REGION AND LANDMARK

Here we describe region and landmark-related features, except those related to propagation directions. Again, **`L`** is the number of landmarks and **`R`** is the number of regions. They are either drawn by the user or generated from the mask builder. The index of the regions and landmarks are labeled on the AQuA GUI. They are also exported as a PNG file.

Notes on the 2D maps in features

In MATLAB, 2D binary images are stored in a 2D array. Assume the image is H by W. The top left element of the matrix has coordinate (1,1), and the bottom left corner of the matrix has coordinate (1,H). On the other hand, the image coordinate system starts from the bottom left, which has a coordinate (1,1). For consistency, in AQuA, when storing 2D images to a matrix, we store coordinate (i,j) in image to (H+1-i,j) in matrix. For example, (1,1) in the image is saved to coordinate (1,H) in the matrix, This makes the matrix look the same as the image if we directly display the image in MATLAB. Also, when using high level display functions in MATLAB (like **`imshow`**), the images are correctly shown. The drawback is that the coordinate systems mismatch. For example, a large Y coordinate actually points to south, instead of north. (for more details, see **https://www.mathworks.com/help/images/image-coordinate-systems.html**)

Field	**Type**	**Size**	**Units**	**Description**
region.landMark.mask	cell	**`L`**		Binary map for each landmark
region.landMark.center	double	**`L`** by 2	pixel	Centroid locations for each landmark
region.landMark.border	cell	**`L`**	pixel	Boundary pixel for each landmark
region.landMark.centerBorderAvgDist	double	**`L`**	µm	Average distance from landmark

				border to its centroid
region.landmarkDist.distPerFrame	cell	**N**	µm	Minimum distance of an event to landmark at each frame
region.landmarkDist.distAvg	double	**N** by **L**	µm	Average minimum distance of an event to each landmark
region.landmarkDist.distMin	double	**N** by **L**	µm	Minimum distance of an event to each landmark
region.cell.mask	cell	**R**		Binary map for each region
region.cell.center	double	**R** by 2	pixel	Centroid locations for each region
region.cell.border	cell	**R**	pixel	Boundary pixel for each region
region.cell.centerBorderAvgDist	double	**R**	µm	Average distance from region border to its centroid
region.cell.incluLmk	double	**R** by **L**		Whether a region contains a landmark
region.cell.memberIdx	double	**N** by **R**		Whether an event is (partially) inside a region
region.cell.dist2border	double	**N** by **R**	µm	Distance of an event to the border of a region
region.cell.dist2borderNorm	double	**N** by **R**	µm	Normalized distance of an event to the border of a region

`region.landMark.mask`

Each element of the cell is a H by W binary matrix showing the location of the landmark. For problems in the coordinate system, see notes above.

`region.landMark.center`

Each row is the center X and Y coordinates for each landmark.

`region.landMark.border`

Each item is the border pixels for each landmark. It is saved as an array, with each row as the X and Y coordinates of a border pixel. It is obtained by **`bwperim`** function in MATLAB.

`region.landMark.centerBorderAvgDist`

For each border pixel, we find the distance between it and its centroid. We report the average distance in this feature. This may be used as a measure for landmark size.

`region.landmarkDist.distPerFrame`

Each item is a T0 by `L` matrix, which is the minimum distance of one event to each landmark at different time points. The range of time is from `loc.t0` to `loc.t1`.

`region.landmarkDist.distAvg`

Each item is the **average minimum** distance of an event to each landmark across all relevant frames. It is obtained by taking the average along time for feature `region.landmarkDist.distPerFrame`.

`region.landmarkDist.distMin`

Each item is the minimum distance of an event to a landmark across all frames when the event exists. It is obtained by taking the minimum along time for feature `region.landmarkDist.distPerFrame`.

`region.cell.mask`

Binary matrix for regions. See also `region.landMark.mask`

`region.cell.center`

Center locations for regions. See also `region.landMark.center`

`region.cell.border`

Boundary pixels for regions. See also `region.landMark.border`

`region.cell.centerBorderAvgDist`

Average distance from centroid to boundaries for regions. See also
`region.landMark.centerBorderAvgDist`

`region.cell.incluLmk`

The relationship between landmarks and regions. If item (i,j) is 1, region i contains landmark j. A region does not need to contain any landmark, and a landmark can be outside any region.

`region.cell.memberIdx`

If item (i,j) is 1, event i is (at least partially) included in regions j. Here we only consider the spatial footprint. Many events are outside any region, so the corresponding row for that event will be all zeros. Sometimes regions can overlap, so some events can belong to multiple regions. Some regions could contain no events.

`region.cell.dist2border`

For each event, we use the spatial footprint (`loc.x2D`) to find the minimum distance to the border of each region, as long as the event (at least partially) is inside the region. Otherwise we use `nan` as distance.

Note

> We do not consider the movement with respect to regions, so we do not record the distance per frame. Use landmark to achieve that goal.

region.cell.dist2borderNorm

We divide values in **region.cell.dist2border** by the corresponding **region.cell.centerBorderAvgDist**. This normalizes the distance to border by the radius of the region.

1.3.5 PROPAGATION DIRECTION (LANDMARK)

It is interesting to study whether the events are propagating toward a landmark, away from a landmark, both, or neither. We save these kinds of features in **res.fts.region.landmarkDir**. The thresholds used in this part is:

```
minThr:0.1:0.9;
```

Here **minThr** is a user-specified parameter. The frames used are from **loc.t0** to **loc.t1** for each event.

Algorithm overview

For each event, under each intensity threshold, we track the change of spatial locations along time. Let A be the set of pixels at time t=0 and B the set at time t=1. The landmark is at location m. Assume the event is growing from time 0 to time 1. We define the new pixels at t=2 as C=B\A, where \ means removing elements (pixels) in A from B. We also define the boundary pixels of A as bd(A).

For each pixel in C, we find the closest pixel in bd(A). The pixel in C has a (geodesic) distance L0 to the landmark, and the pixel in bd(A) has (geodesic) distance L1 to the landmark. If L1 is smaller than L0, the pixel implies propagating toward the landmark. Some pixels in C have an L1 larger than L0, which means they imply propagating away from the landmark.

We can gather all pixels in C with L1 smaller than L0 to get the score of propagating toward that landmark at this frame. Similarly, we have the score of propagating away from the landmark by gathering the pixels in C with L1 larger than L0. If the pixels in C have larger changes between L0 and L1, and/or there are a lot of pixels in C, we will have larger scores.

Unlike the features in single event propagation, we do not consider shrinking. In other words, if at t=0 the pixels have intensity higher than the threshold, but at t=1 the intensity becomes lower than the threshold, we do not consider this as propagating toward or away from the landmark.

Additionally, the scores here are not normalized. Therefore, larger events with longer durations are more likely to have larger scores. Some ways to compensate this are provided in the overlay module of AQuA.

Notes: units

The unit for the score of individual pixels is the length (here we use μm). When the events propagate, we gather the score from pixels covering some area. Therefore, when we take the summation of the scores among pixels, the results can be interpreted as volumes. So, we use the unit μm^3. We can also interpret this summation as the addition of length, therefore the units can be just μm^3.

Notes: disconnected components

At t=0, the events have several disconnected spatial components that do not connect with each other. At t=1, the events grow and these components merge. Then how do we determine whether the events are propagating toward or away from the landmark? For each new pixel in `C`, we need to determine the corresponding pixel in bd(A). When doing this, we can use the component sizes of A as a weight. A component with a larger size in A will have a larger weight and becomes more likely to provide a pixel for the pixel in C. This arrangement is more consistent with visual perception. As an extreme case, assume we have two components in t=0, one is very large, and one is very small, then at t=1, if these two merge, we will have the impression that the newly grown pixels (C) mainly come from the larger component. In the algorithm, if the size a component in A is smaller than 20% of C, we ignore that component when calculating the scores.

Notes: before and after reaching the landmark

At t=0, the event has not reached the landmark yet, at t=1, the event covers the landmark. Therefore, from t=0 to t=1, the event is both propagating toward and away from the landmark. For example, at t=0.5, the frontier of the event has just hit the landmark. Then, from t=0.5 to t=1, the frontier continues moving and winds up far away from the landmark. When calculating the score for individual pixels, the algorithm first checks whether the frontier has ever hit the landmark, which leads to distance 0. If it is the case, we will report both propagating toward and away from the landmark.

Field	**Type**	**Size**	**Units**	**Description**
region.landmarkDir.chgToward	double	`N` by `L`	µm^3	Score of an event propagating toward each landmark
region.landmarkDir.chgAway	double	`N` by `L`	µm^3	Score of an event propagating away from each landmark
region.landmarkDir.chgTowardBefReach	double	`N` by `L`	µm^3	Score of an event propagating toward each landmark before reaching it
region.landmarkDir.chgAwayAftReach	double	`N` by `L`	µm^3	Score of an event propagating away from each landmark after reaching it
region.landmarkDir.pixelToward	cell	`N`	µm	Score of each pixel in an event propagating toward each landmark
region.landmarkDir.pixelAway	cell	`N`	µm	Score of each pixel in an event propagating away from each landmark
region.landmarkDir.chgTowardThr	double	`N` by `Thr` by `L`	µm^3	Score of an event propagating toward each landmark with given threshold
region.landmarkDir.chgAwayThr	double	`N` by `Thr` by `L`	µm^3	Score of an event propagating away from each landmark with given threshold

`region.landmarkDir.chgToward`

The score for each event propagating toward each landmark.

`region.landmarkDir.chgAway`

The score for each event propagating away from each landmark.

`region.landmarkDir.chgTowardBefReach`

The score of each event propagating toward each landmark. But we do not consider the scores after reaching the corresponding landmark. This feature, along with **`region.landmarkDir.chgAwayAftReach`** is useful if we want to distinguish the propagation pattern before and after reaching the landmark.

`region.landmarkDir.chgAwayAftReach`

The score of each event propagating away from each landmark. But we do not consider the scores before reaching the corresponding landmark. This feature, along with **`region.landmarkDir.chgTowardBefReach`** is useful if we want to distinguish the propagation pattern before and after reaching the landmark.

`region.landmarkDir.pixelToward`

For each event, there is a H by W by L array. Let **`x=res.fts.region.landmarkDir.pixelToward`**, then **`x(:,:,l)`** is a 2D array for the *l*th landmark. The array saves the score of propagating toward that landmark for each pixel.

`region.landmarkDir.pixelAway`

For each event, there is a H by W by L array. Let **`x=res.fts.region.landmarkDir.pixelAway`**, then **`x(:,:,l)`** is a 2D array for the *l*th landmark. The array saves the score of propagating away from that landmark for each pixel.

`region.landmarkDir.chgTowardThr`

Similar to **`region.landmarkDir.chgToward`**, except that the scores for all thresholds are provided.

`region.landmarkDir.chgAwayThr`

Similar to **`region.landmarkDir.chgAway`**, except that the scores for all thresholds are provided.

1.3.6 EVENT GROUPS

Features related to more than one event are put under **`res.fts.network`**.

Field	**Type**	**Size**	**Units**	**Description**
networkAll.nOccurSameLoc	double	N by 2		Number of events whose spatial footprint overlaps with current one
networkAll.nOccurSameTime	double	N		Number of events that occur at the same time as the current one
networkAll.occurSameLocList	cell	N by 2		List of events whose spatial footprint overlaps with the current one

networkAll.occurSameTimeList	cell	N		List of events that occur at the same time as the current one
network.nOccurSameLoc	double	N by 2		Only use events inside regions. Number of events whose spatial footprint overlaps with the current one
network.nOccurSameTime	double	N		Only use events inside regions. Number of events that occurs at the same time as the current one
network.occurSameLocList	cell	N by 2		Only use events inside regions. List of events whose spatial footprint overlaps with the current one
network.occurSameTimeList	cell	N		Only use events inside regions. List of events that occurs at the same time as the current one

`networkAll.nOccurSameLoc`, `networkAll.occurSameLocList`

For event i, we find the events whose spatial footprint overlaps with it. We count the number and save it in element **`(i,1)`** of **`nOccurSameLoc`**. The list of events is saved in item **`(i,1)`** of **`occurSameLocList`**. Then from the list we find those whose size is not too different from event i (from 50% to 200%) and save the list in **`(i,2)`** item in **`occurSameLocList`**. We count the number again and save it in element **`(i,2)`** of **`nOccurSameLoc`**. In all cases, event i itself is counted.

`networkAll.nOccurSameTime`, `networkAll.occurSameTimeList`

Assume event i starts at frame t0 and ends at frame t1, then we find all events that exists during t0 and t1. The list is saved in **`nOccurSameTime`** and the number is saved in item i in **`occurSameTimeList`**.

`network.nOccurSameLoc`, `network.occurSameLocList`

The same as **`networkAll.nOccurSameLoc`** and **`networkAll.occurSameLocList`**, except that we only use events inside the regions.

`network.nOccurSameTime`, `network.occurSameTimeList`

The same as **`networkAll.nOccurSameTime`** and **`networkAll.occurSameTimeList`**, except that we only use events inside the regions.

1.3.7 DERIVED FEATURES

Some features in the exported file and the overlay function is derived from the above features. Some variables are defined below. The description can be found in the overview of outputs.

`x`

The output of the calculation for each feature.

`fts`

Feature data structure: either `res.fts` or `res.ftsFilter`.

`nEvt`

Event number.

`xxDi`

Propagation direction of interest. 1. Anterior. 2. Posterior. 3. Lateral left. 4. Medial right

`xxLmk`

Landmark index of interest. The number is the same as those annotated in the overlay.

Name	**Script**
Index	x=1:nEvt
Basic - Area	x=fts.basic.area
Basic - Perimeter	x=fts.basic.peri
Basic - Circularity	x=fts.basic.circMetric
Curve - P Value on max Dff (-log10)	x=-log10(fts.curve.dffMaxPval)
Curve - Max Dff	x=fts.curve.dffMax
Curve - Duration 50% to 50%	x=fts.curve.width55
Curve - Duration 10% to 10%	x=fts.curve.width11
Curve - Rising duration 10% to 90%	x=fts.curve.rise19
Curve - Decaying duration 90% to 10%	x=fts.curve.fall91
Curve - Decay tau	x=fts.curve.decayTau
Propagation - onset - overall	x0=fts.propagation.propGrowOverall; x=sum(x0,2)
Propagation - onset - one direction	x0=fts.propagation.propGrowOverall(:,xxDi); x=x0
Propagation - onset - one direction - ratio	x0=fts.propagation.propGrowOverall; xAll=sum(x0,2); xSel=x0(:,xxDi); x=xSel./xAll
Propagation - offset - overall	x0=fts.propagation.propShrinkOverall; x=sum(abs(x0),2)
Propagation - offset - one direction	x0=fts.propagation.propShrinkOverall(:,xxDi); x=abs(x0)
Propagation - offset - one direction - ratio	x0=abs(fts.propagation.propShrinkOverall); xAll=sum(x0,2); xSel=x0(:,xxDi); x=xSel./xAll

Landmark - event average distance	x=fts.region.landmarkDist.distAvg(:,xxLmk)
Landmark - event minimum distance	x=fts.region.landmarkDist.distMin(:,xxLmk)
Landmark - event toward landmark	x=fts.region.landmarkDir.chgToward(:,xxLmk)
Landmark - event away from landmark	x=fts.region.landmarkDir.chgAway(:,xxLmk)
Landmark - event toward landmark before reaching	x=fts.region.landmarkDir.chgTowardBefReach(:,xxLmk)
Landmark - event away from landmark after reaching	x=fts.region.landmarkDir.chgAwayAftReach(:,xxLmk)
Region - event centroid distance to border	x0=fts.region.cell.dist2border;x=nanmin(x0,[],2)
Region - event centroid distance to border - normalized by region radius	x0=fts.region.cell.dist2borderNorm; x=nanmin(x0,[],2)
Network - Temporal density	x=fts.network.nOccurSameLoc(:,1)
Network - Temporal density with similar size only	x=fts.network.nOccurSameLoc(:,2)
Network - Spatial density	x=fts.network.nOccurSameTime

2 PARAMETERS GUIDE

If you run into a problem, go through this part to help you choose optimal parameters and obtain suitable results. A complete parameter list is also provided.

2.1 PREPROCESSING, PARAMETER TUNING, AND TROUBLESHOOTING

2.1.1 PRE-PROCESSING

AQuA requires the data quality to be reasonably good. For example, if there are strong uncorrected motion artifacts or bright light noise, the performance will degrade. Below are some commonly used pre-processing steps to consider for some data sets. AQuA supports gray-scale time-lapse TIFF data with 8, 16 or 32 bits of depth.

Alignment/motion correction should be performed as needed before AQuA can be used. This can be done with an existing tool, for example, NoRMCorre (https://github.com/flatironinstitute/NoRMCorre).

If you observe grainy noise that looks like salt and pepper and AQuA does not work well with moderate parameter tuning, please consider applying 2D or 3D median filtering before sending to AQuA. In MATLAB, the median filter function is medfilt2. AQuA is designed to handle Poisson-Gaussian noise or pure Gaussian noise. More details can be found **here**.

If your data contains strong light noise, you can first try AQuA without any preprocessing, but you need to set a stronger smoothing level (see signal panel part below). If that still cannot distinguish the true signal from noise, consider using median filtering on the data.

If you find severe background trends (for example, strong and quick bleaching), please consider correcting it first. AQuA can handle mild photo-bleaching trends, though. You can first try to directly apply PCA (principal component analysis) or NMF (non-negative matrix factorization) and remove the top 1 or 2 components which usually corresponds to the trend. If that does not work well (e.g., true signals are lost), please try the sample script provided with AQuA (correct_baseline.m). The script first detects events, then estimates the background trend after removing those events. This may potentially give better results.

2.1.2 OPEN FILE WINDOW

The names before the dashed line are those shown in the GUI, those after are the parameter names in presets.

Preset types

You need to choose a preset that best matches your data set. Each preset has different choices of parameters. You can add or modify presets in ‘parameters1.csv’ in the ‘cfg’ folder of AQuA.

Temporal resolution – frameRate

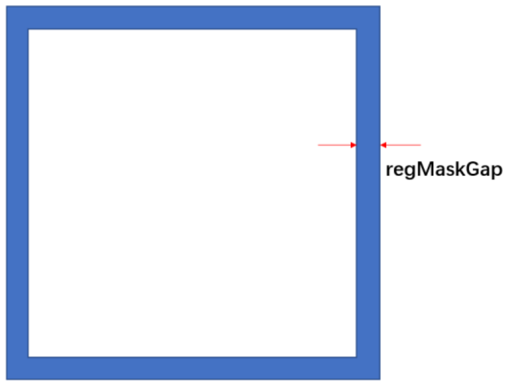
The unit is second per frame. This will be applied when features are calculated. This will not influence the detection algorithm.

Spatial resolution – spatialRes

The unit is micrometer per pixel. This will be applied when features are calculated. This will not influence the detection algorithm.

Exclude pixels – regMaskGap

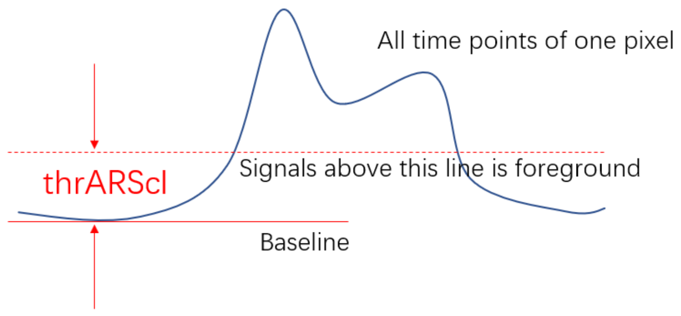
If you apply motion correction to the movie, there could some black lines or regions on the boundary of the movie. They need to be avoided before further processing. For example, if your movie has a field of view of 512 by 512 pixels, setting this parameter to 5 will remove pixels whose distance is smaller or equal to 5 pixels from the border.



2.1.3 SIGNAL TAB

Intensity threshold scaling factor – thrARScl

First, we estimate the noise level of the movie. Assume the standard deviation of the noise is σ. Then, for each pixel in each time point, if the delta F values are higher than $thrARScl \times \sigma$, we mark that pixel at that frame as active. In that frame, events will be limited to those active pixels. Typically, the values can range from 2 to 3.


All time points of one pixel
thrARScl
Signals above this line is foreground
Baseline

Smoothing (sigma) – smoXY

We smooth the movie spatially using a Gaussian filter with a standard deviation of smoXY. For a movie with less noise, we prefer a smaller value to preserve more details. Typically, we choose around 0.5. For movies with higher noise, a larger value is needed. We can choose around 1. For very noisy movies, we may try 1.5 or 2, but these kinds of large values also distort the shape of events.

Minimum size (pixels) – minSize

After smoothing with smoXY and thresholding with thrAR, in each frame, we obtain some connected regions in the foreground. If the size of a component is only one or two pixels, it is likely due to noise. Therefore, all components whose sizes are smaller than minSize will be discarded. Typically, we choose 8 when smoXY is 0.5. If smoXY is 1, it is safer to choose minSize as 16 because stronger smoothing also tends to lead to more artifacts. If you already have prior knowledge about the size of the events, you can also choose an even larger value for minSize. Though we recommend to set a smaller value here and use the proof-reading tool to filter out those smaller and/or darker ones after detection.

Troubleshooting

If you detected too few events, or the size of events is too small, try decreasing thrAR. If you find decreasing thrAR will include too much noise, consider increasing smoXY and minSize. If you find too many events are detected, first increase thrAR. If it does not work, also try increasing smoXY and minSize.

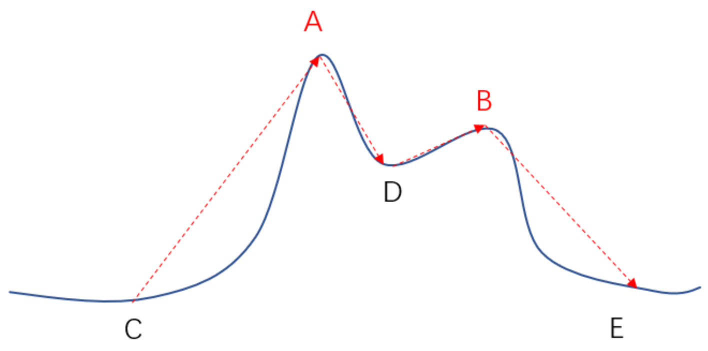
If you find you need to choose a very large thrAR in order to get the signals detected, it is likely that the noise standard deviation is severely underestimated. In this case, consider using a median filter to do some pre-processing (see pre-processing section above).

2.1.4 VOXEL TAB

Temporal cut threshold - thrTWScl

If a spatial location in the movie has several events, we use a 'temporal cut' threshold to determine single events. For example, say an area's signal increases, then decreases, then increases again and finally decreases. If the first decrease and the second increase is strong enough, we feel there are two events in that location and we make a 'temporal cut' between these two events. If either of them is weak, it is likely there is only one event. This parameter defines the threshold for being 'weak'. The typical value is 2.

Based on the foreground signals detected in the signal panel, we find foreground voxels that are stronger than their spatial and temporal neighbors. We call them seeds. Each seed corresponds to a curve if we draw the intensity values over time for that spatial location. We want to achieve two goals. First, we assume each seed is part of an event, so we want to determine, for that seed, when the event starts and when it ends. Second, if the curve contains multiple peaks, we need to determine whether these peaks belong to the same event, or if we should cut them into multiple events.



We use the example above to illustrate how we achieve these two goals. Define Δ=thrTWScl×Noise σ. There are two seeds: A and B. If A-C and A-D are both larger than Δ, A is the peak of an event. Similarly, if B-D and B-

E are both larger than Δ, B is the peak of an event. If A-C and B-E are larger than Δ, but B-D or A-D is not, A and B will be considered as one event. If none of them are larger than Δ, then there are no events.

Growing z threshold - thrExtZ

We already assume each seed belongs to an event and find a rough duration for that seed using thrARScl. Now we need to determine whether the neighboring pixels surrounding each seed are similar enough to be included in that seed. For 'similar,' we mean the co-occurring signal has a similar onset and offset pattern. If the pattern is similar enough, we say those pixels are also members of that event. We use thrExtZ to define the threshold of 'similar enough'. This value is related to the noise level. The typical value is 1.

Troubleshooting

If we prefer to make fewer temporal cuts, try larger values like 4. Setting values even larger usually disables temporal cut.

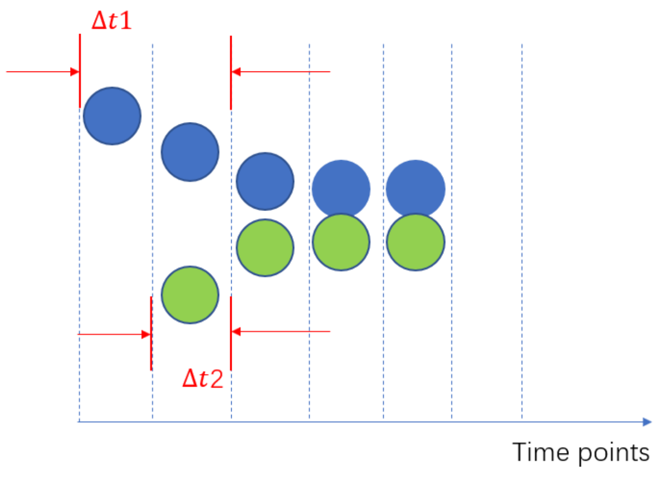
If we feel that too many pixels are considered similar to the seeds, which makes the color-coded areas too large, consider increasing thrExtZ (e.g., 2). On the other hand, decrease thrExtZ (e.g., 0.5 if too many signals are not included.

2.1.5 EVENT TAB

After the voxels step, we assign pixels to seeds. The seed along with the neighboring pixels that belong to it is a super voxel. Each event could contain multiple super voxels. We first combine super voxels to super events using the parameter cDelay. Then we segment each super event to events using cRise and cDelay. The smoothness of the estimated propagation for each event is controlled by gtwSmo.

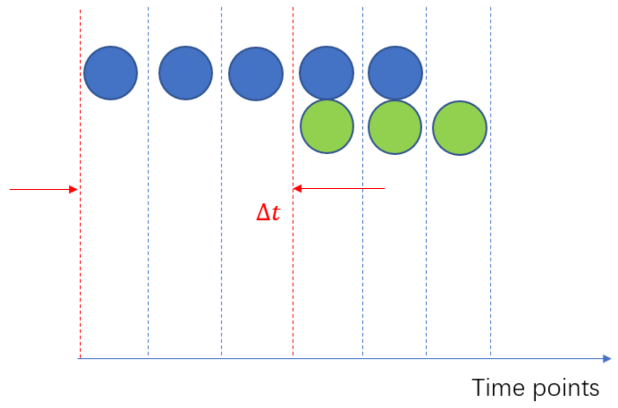
Rising time uncertainty - cRise

For a given super event, we may find that several pixels become brighter earlier than other pixels. It appears as if the signals start growing from these pixels and meet together later. We use cRise to determine whether each of such pixels is unique enough to be considered as the starting point of an event. The typical value of cRise is 1 or 2 frames. The meaning of this parameter is best depicted in the example below. There is a super event that lasts for five-time points. If $\Delta t1$<cRise **or** $\Delta t2$<cRise, green and blue parts are considered as one event. Otherwise, the super event contains only one event. If Δt1 or $\Delta t2$ is too small, it is likely due to the uncertainty of the movie and/or the rising time estimation.



Slowest delay in propagation - cDelay

If the co-occurring time of two neighboring parts is very short, we think they do not belong to the same event and/or super event. The maximum onset time difference is controlled by cDelay. A smaller value will lead to smoother propagation within an event and/or super event. The typical value is 1 or 2 frames. In the example below, if Δt<cDelay, green and blue parts are considered as one event.



Propagation smoothness - gtwSmo

We use graphical time warping to get the propagation patterns of events. The smooth parameter determines how flexible the propagation can be. This value is also related to noise levels. The default value is 1, but can range from 0.5 to 4 in typical settings.

Troubleshooting

The cRise and cDelay will have a major impact on the number and size of events. If you found the number is too small, decrease cRise and cDelay. For example, try cRise with 0.5 and cDelay also with 0.5. Otherwise, increase them. For example, set cRise to 5 and cDelay to 5.

If you find the propagation path looks too noisy, increase gtwSmo, like 4 or 8. If too many details are lost, try smaller values like 0.5 or even 0.1.

If in this step, you find events are missed, go back to earlier steps to make sure the signals are detected.

2.1.6 CLEAN TAB

Z score threshold - zThr

We calculate a z-score for each detected event. If the score is very low, then we cannot distinguish them from noise. This step removes events whose z-scores are lower than zThr. The main purpose for this step is to prevent these events from interfering with the merging step. If you do not use merging, we suggest a smaller value of 2 and to perform filtering in the proof-reading tool. Otherwise, you may optionally try a larger value, though sometimes a smaller value still works better.

Troubleshooting

Please set it smaller (like 0.5 or 1) if too many events are removed. If a lot of noise is not removed, set it larger (about 5, or even larger, depending on the type of noise). Again, if you do not use the merging function, we prefer a smaller value in this step.

2.1.7 MERGE TAB

Ignore merging - ignoreMerge

The following parameters will take effect only when this parameter is zero, which will uncheck the checkbox. Merging operation is used when you think an event could contain spatially unconnected components. By default, this parameter is set to 1.

Maximum distance - mergeEventDiscon

Two events detected in previous steps whose distances are larger than mergeEventDiscon will not be merged.

Minimum correlation - mergeEventCorr

We calculate the correlation of curves between events detected. If the correlation between the two events is below mergeEventDiscon, they will not be merged. The values should be between 0 and 1.

Maximum time difference - mergeEventMaxTimeDif

We calculate the onset time of each event detected. If the onset time of two events is larger than mergeEventMaxTimeDif, they will not be combined.

Troubleshooting

If too many events are merged, try increasing mergeEventDiscon, increase mergeEventCorr and decrease mergeEventMaxTimeDif.

2.1.8 RECON TAB

This step re-detects events if the merging operation is not ignored. There is no parameter to choose.

2.1.9 FEATURE TAB

This step extracts the features.

Ignore decay tau – ignoreTau

Set this to 1 if you do not need the decay time constant as an output feature.

2.1.10 MORE PARAMETERS

Some other parameters are occasionally used as well. They are not so frequently encountered as those in the detection pipeline tabs, so you need to change them in the preset file.

cut

Before detection, AQuA cuts the movie to several parts, each part contains ‘cut’ sub-stacks. For a movie whose background is changing quickly, consider choosing a smaller cut (like 100). Otherwise, it is better to use a larger value (like 400). Using smaller values may potentially lead to problems for events with longer durations.

movAvgWin

This parameter specifies the number of frames used to calculate the baseline level. We use a moving average filter whose length is movAvgWin to process the movie. If the movie contains a long period without a signal, a larger movAvgWin (like 100) makes the baseline more accurate. This is usually the case for in vivo data sets. Otherwise, a smaller movAvgWin (like 25) is more suitable for very active data sets, like some single cell data sets.

superEventdensityFirst

We use different strategies to combine super voxels to super events. If superEventdensityFirst is set to 1, we start combining from the time point with the largest number of super voxels. This is more suitable for in vivo data sets. If it is set 0, we start from the earliest onset super voxel, which is more suitable for most single cell ex vivo data sets.

minShow1

For each event, the signals lower than a certain percentage of the peak ΔF will be discarded. The default value is 0.2. For noisy data or data with residual trends, you may want a higher value. For less noisy data, a smaller value like 0.1 could be better.

correctTrend

Given an intensity curve of a detected event, we may want to remove the baseline trend due to effects like photo-bleaching. We first remove the time points when the events occurred in that curve. Then we fit the remaining time points with a polynomial curve whose order is correctTrend. Then we subtract the fitted curve from the intensity curve and calculate the $\Delta F/F$ curve. If correctTrend is 1 (default), we use a line to fit. Usually, this works well enough. For data with a more complicated trend, consider setting it to 2. For data without any obvious trend, set it to 0 to avoid any potential distortion introduced by trend correction.

2.2 PARAMETERS LIST

Each step in the pipeline involves several parameters. Most of them do not need to be changed.

2.2.1 DATA PARAMETERS

Name	**Variable**	**Type**	**Value**	**Notes**
Temporal resolution	frameRate	prep	0.5	Seconds per frame
Spatial resolution	spatialRes	prep	0.5	Micrometers per pixel
Estimated noise variance	varEst	prep	0.02	
Foreground threshold	fgFluo	prep	0	
Background threshold	bgFluo	prep	0	
X cooridante for north vector	Northx	prep	0	

Y cooridante for north vector	Northy	prep	1	

2.2.2 MAIN PARAMETERS

Name	**Variable**	**Type**	**Value**	**Notes**
Minimum size	minSize	prep	8	Minimum size of events to be detected (in pixels)
Spatial smoothing level	smoXY	prep	0.5	Spatial smoothing filter size
Active voxels threshold scale	thrARScl	prep	2	
Temporal cut threshold scale	thrTWScl	prep	2	
Seed growing threshold	thrExtZ	prep	1	
Slowest propgation	cDelay	event	2	
Rising phase uncertainty	cRise	event	2	
GTW smoothness term	gtwSmo	event	1	
GTW windows size	maxStp	event	11	Maximum propagation delay
Z score threshold for events	zThr	post	2	
Ignore merging step	ignoreMerge	post	1	
Maximum merging distance	mergeEventDiscon	post	0	
Minimum merging correlation	mergeEventCorr	post	0	
Maximum merging time difference	mergeEventMaxTimeDif	post	2	

2.2.3 EXTRA PARAMETERS

Name	**Variable**	**Type**	**Value**	**Notes**
Remove pixels close to image boundary	regMaskGap	prep	5	

Poisson noise model	usePG	prep	1	
Frames per segment	cut	prep	200	Cut video to pieces to avoid global trend
Baseline window	movAvgWin	prep	25	
Extend super voxels temporally	extendSV	prep	1	
Older code for active voxels	legacyModeActRun	prep	1	
Time window detection range	getTimeWindowExt	prep	50	Smaller value for faster run
Pixels for window detection	seedNeib	prep	1	
Remove seeds	seedRemoveNeib	prep	2	Should be >= seedNeib
Super voxel significance	thrSvSig	prep	4	
Super events prefer larger	superEventdensityFirst	event	1	
Area ratio to find seed curve	gtwGapSeedRatio	event	4	Larger for smaller proportion
Area to find seed curve	gtwGapSeedMin	event	5	
Spatial overlap threshold	cOver	event	0.2	
Event show threshold on raw data	minShow1	post	0.2	
GUI event boundary threshold	minShowEvtGUI	post	0	Smaller to show boundary longer
Ignore decay tau calculation	ignoreTau	post	1	
Correct baseline trend	correctTrend	post	1	The polynomial order of the model to correct baseline trend
Extend event temporally after merging	extendEvtRe	post	0	