QuimP Guide

Piotr Baniukiewicz, Richard Tyson and Till Bretschneider
Department of Computer Science, Warwick University
Correspondence to Till Bretschneider. Email: Till.Bretschneider@warwick.ac.uk

Version: 19.04.02

Web site: http://www.warwick.ac.uk/quimp

This manual is available on GitHub. Feel free to fork it and make pull requests!
https://github.com/CellDynamics/QuimP-Doc

Manual compiled on Friday 26th April, 2019 at 07:39

Contents

1 License and citation

The full text of QuimP license is available at https://pilip.lnx.warwick.ac.uk/LICENSE.txt.

Please, make the following citation in any publication related to our software:

"QuimP [1] used in this study was developed at the University of Warwick with support from BBSRC (BBR grant BB/M01150X/1).

[1] Piotr Baniukiewicz, Sharon Collier, Till Bretschneider QuimP: analyzing transmembrane signalling in highly deformable cells. *Bioinformatics*, Volume 34, Issue 15, 1 August 2018, Pages 2695-2697, doi: 10.1093/bioinformatics/bty169

2 PDFs and online versions of this manual

The user manual exists in two versions, *final* and *development*. The final documentation is (usually) related to the current official release of QuimP, whereas the development version evolves together with the software, following upcoming features, bugfixes, etc.

The most recent documentation can be downloaded from:

1. **Final documentation**
   - PDF - https://pilip.lnx.warwick.ac.uk/docs/master/QuimP_Guide.pdf
   - HTML - https://pilip.lnx.warwick.ac.uk/docs/master/QuimP_Guide.html
2. **Development documentation**
   - PDF - https://pilip.lnx.warwick.ac.uk/docs/develop/QuimP_Guide.pdf
   - HTML - https://pilip.lnx.warwick.ac.uk/docs/develop/QuimP_Guide.html

3 About QuimP

- QuimP software for analysing cellular morphodynamics has been developed on and off since 2002, and was first described in Dormann D, Libotte T, Weijer CJ, Bretschneider T. Simultaneous quantification of cell motility and protein-membrane-association using active contours. Cell Motil Cytoskeleton. 2002 Aug;52(4):221-30. doi: 10.1002/cm.10048.
- Since then it has contributed to about more than 80 publications.
- QuimP development is currently funded through BBSRC BBR grant BB/M01150X/1: 'QuimP software for quantifying cellular morphodynamics' (PI Bretscheider), Jul 2015-Dec 2018, with Piotr Baniukiewicz as main developer.
- Current development aims at making QuimP a sustainable resource for the biomedical community. This involves releasing QuimP as open source for academic use, creating a number of web tools for collaborative development and improved interaction with users.

4 QuimP - what is new

4.1 Summary

The key features of releases:

1. v19.04.01
   - Added Omero import/export option (subsection 7.1).
2. v19.03.01
   - Reworked Protrusion Tracking module (section 13).
   - Python examples of processing *QCONF* at GitHub account[1]
3. v18.10.01
   - Optimization of Random Walk module for multi-threading execution (subsection 12.2).
4. v18.02.01
   - Enhanced macro support. The whole workflow can be run as macro - see subsection 6.2
   - New module for generating Qconf files from segmented images. It can replace BOA if one has already segmented image - see subsection 8.5
5. v18.01.01
   - Macro support - see section 15
   - Save BOA state without quitting - see section 8, subsection 8.1
   - Added option in BOA for expanding snakes from cell inside - see section 8
   - Significant changes in Random Walk module, e.g. multi object segmentation - see subsection 12.2
6. v17.12.02
   - Freeze cells - see section 8, subsection 8.2, subsection 8.1
   - Median Snake filter - see subsection 8.3
   - Random Walk plugin - relative error setting, see subsection 12.2
7. New QuimP (after QuimP11)
   - The ability to save and restore the state of the BOA plugin for cell contour segmentation. Module configuration, plugin settings and current view are saved in the new file format *QCONF*. To maintain backward compatibility, files in the previous format are can be still generated (refer to section 14). The *QCONF* file is human readable using the JSON format and contains all data and parameters generated by BOA. This format is also used by other QuimP modules such as ECMM and Q Analysis.

[1] https://github.com/CellDynamics/QuimP-Python

- New image preprocessing plugins:
  - DIC image reconstruction
  - Random Walker segmentation
  - Protrusion analysis
- The BOA plugin can use binary masks generated outside QuimP to create cell outlines (Snakes).

4.2 Changelog

A changelog can be found at the QuimP web page at: `https://pilip.lnx.warwick.ac.uk/site/changes-report.html`. Version of QuimP you are running as well as the version history can be viewed directly from the QuimP Bar plugin (Figure 1).

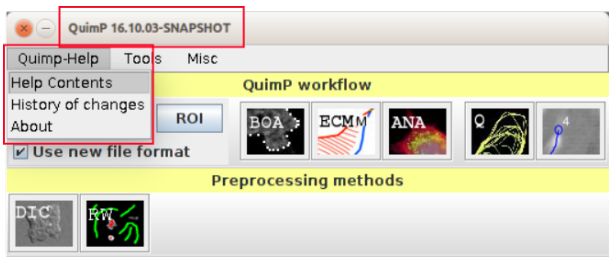


Figure 1: The QuimP bar

If you find a bug or have specific feature requests, please contact the QuimP developer: p.baniukiewicz@warwick.ac.uk

5 Installation

Consult our Wiki page for installation details:
http://warwick.ac.uk/quimp/wiki-pages/

6 Quick start

Matlab routines and walkthrough data mentioned in this section are available from QuimP web page[2].

Check also our latest publication: Piotr Baniukiewicz, Sharon Collier, Till Bretschneider; QuimP Analyzing transmembrane signalling in highly deformable cells, *Bioinformatics*, which contains many examples and workflows in supplementary materials.

Simple example of processing QuimP datafiles in Python is available at our GitHub account[3]

6.1 Walkthrough analysis, example 1

The folder *QuimP_walkthrough_files* contains 3 tiff image sequences which you can analyse (images are courtesy of Evgeny Zatulovskiy, Rob Kay Group, MRC Laboratory of Molecular Biology).

You can find the results from above analysis in *example_output* folder, exemplary post-processing in Matlab are available for viewing here[4].

- *QW_channel_1_actin.tif*- Actin label. Image and has been background corrected and contrast enhanced.
- *QW_channel_2_neg.tif* - Negative stain. The cell appears as a shadow on a bright background, which has then been inverted.
- *QW_channel_2_seg.tif* - For segmentation we shall use the negative stain channel. A 1 pixel Gaussian blur has been applied, the background removed, and contrast enhanced.
- *Segmentation.tif* - Already segmented image - will not be used in this tutorial.

Step 1. Open ImageJ and launch the QuimP bar [*Plugins* → *QuimP* → *QuimPBar*]. Open the image *QW_channel_2_seg.tif*. Launch BOA from the QuimP bar.

Step 2. At the prompt check the scale is correct (2 second frame interval, pixel width 0.2 microns).

Step 3. Using the polygon selection tool, create a selection encompassing the cell. This can be very rough. Click *Add cell*.

Step 4. Adjust the parameters to get a good segmentation. Click *SEGMENT* and wait for completion. You can also restore experiment configuration from *QW_channel_2_seg.QCONF* file from *example_output* folder.

Step 5. Scroll through the sequence to check the segmentation result using either the scroll bar or mouse wheel. The segmentation should have completed to the last frame. If not, we will truncate the

[2]http://www2.warwick.ac.uk/fac/sci/dcs/people/till_bretschneider/quimp/test_data
[3]https://github.com/CellDynamics/QuimP-Python/blob/master/qconf_examples.ipynb
[4]https://pilip.lnx.warwick.ac.uk/docs/Walkthrough.html

segmentation. Scroll to the last frame which successfully segmented and click *Truncate Seg*. Click *FINISH*, and provide a name for the analysis (e.g. 'QTest').

Step 6. Launch the ECMM plugin from the QuimP bar. It does not require an image. When prompted, locate the QuimP parameter file you just created (e.g QTest.paQP). ECMM will run. You can view the result and close the image.

Step 7. Open the image *QW_channel_1_actin.tif*. With it in focus, launch the ANA plugin, and again select your parameter file. Choose a sensible cortex width (e.g. 1.5 microns). Make sure 'save in channel' is set to 1, and normalise to interior is ticked. Click *ok*. When complete, ANA will show the sample locations for the last frame.

Step 8. Close *QW_channel_1_actin.tif*, and open *QW_channel_2_neg.tif*. We will record another channel for good measure. Repeat the last step, but with the channel 2 image, this time storing data in channel 2 (which is the default), and tick *Sample at Ch1 locations*. ANA will now use the same sample points as computed for channel 1 (useful if you want to compute ratios between channels).

Step 9. Launch the Q Analysis plugin, and again choose your parameter file. Check your scale is what you expect at the top of the parameter window. We will use all the current defaults, so click *RUN*. Inspect your maps, all of which are automatically saved to disk.

Step 10. The displayed images have been scaled to cover the colour space. The raw values have been saved as text file, with the extension *.maQP* (e.g. *QTest_0_fluoCh1.maQP*). You can view them in ImageJ by opening them via [*File*− > *Import*− > *TextImage*...]. Note that latest versions of QuimP store these files inside *QCONF* configuration file. One can retrieve them by using Format Converter (see section 14).

Step 11 We will now switch to MATLAB to plot some of this data. Open Matlab and open the script file *walkthrough.m*, which will guide you through loading and plotting data. You can use the output provided in the QuimP package, called *QWalkthrough*, rather than your own.

6.2 Walkthrough analysis, macro example

In this example you will perform analysis similar to subsection 6.1 but with help of ImageJ macro. Note that this approach requires already segmented image provided as binary or grayscale mask (we use *Segmentation.tif* from walkthrough files). More about converting masks to QuimP format is given in subsection 8.5. Below you can find exemplary macro that performs the same anaysis like that described in subsection 6.1

```
/**
* Exemplary macro running full QuimP analysis.
*
* Require already segmented image.
*
* Modify paths before use.
*/

// define where to save main configuration file. It will be shared among QuimP modules.
// Any other file generated by QuimP will be saved in this folder as well
```

```
qconfOutput = "experiment.QCONF"
// open segmented image, you can use any other segmentation software to obtain masks
open("Segmentation.tif")
// open original image
open("QW_channel_1_actin.tif")

// 1) perform conversion from mask to QCONF file. This step corresponds to saving segmentation in BOA
run("Generate Qconf", "opts={options:{" +
        // name of the mask image (nod ID)
        "select_mask:Segmentation.tif," +
        // name of the image or path here
        "select_original:QW_channel_1_actin.tif," +
        // density of nodes, step=1 means each pixel of mask will be mapped to node
        "step:4.0,smoothing:true},," +
        // alternatively path to mask file
        "maskFileName:(),," +
        "paramFile:("+qconfOutput+")}");

// 2) run ECMM analysis on configuration file
run("ECMM Mapping", "opts={paramFile:("+qconfOutput+")}");

// 3) run ANA analysis, we use only one channel
open("QW_channel_1_actin.tif"); // image for intensity sampling
selectWindow("QW_channel_1_actin.tif");
run("ANA", "opts={plotOutlines:true,fluoResultTable:true,fluoResultTableAppend:false," +
        // configure displaying
        "channel:0, userScale:5.0," +
        // set channel and cortex width (in um, pixel size from image will be used)
        "normalise:true, sampleAtSame:false," +
        "clearFlu:false," +
        "paramFile:("+qconfOutput+")}");

// 4) run Q analysis
run("QuimP Analysis", "opts={trackColor:Summer," +
"outlinePlot:Speed," +
"sumCov:1.0,avgCov:0.0," +
"mapRes:400," +
"paramFile:("+qconfOutput+")}");

// 5) convert data to csv files and generate coordinates maps
run("Format converter", +
        "opts={status:[ecmm:outlines,ecmm:centroid,map:coord,map:origin,map:ycoords,map:xcoords]," +
        "areMultipleFiles:true," +
        "paramFile:("+qconfOutput+")}");
```

For latest and actual version of macro check walkthrough data archive [5].

[5] http://www2.warwick.ac.uk/fac/sci/dcs/people/till_bretschneider/quimp/test_data

7 QuimP Workflow

There are four stages which are usually performed when tracking cells using QuimP, each handled by a different plugin (Figure 2).

1. Cell segmentation - Performed by the **BOA** plugin. Cell outlines, represented as a chain of nodes, are extracted from each frame of an image sequence using an active contour. BOA outputs global measures, such as cell centroid displacement, or cell area.
2. Membrane tracking - Performed by the **ECMM Mapping** plugin. Outlines are mapped between frames to extract local membrane velocities.
3. Measuring Fluorescence - Performed by the **ANA** plugin. Pixel intensities are sampled from the cell's cortex, its width defined by the user.
4. Data analysis - Performed by the **Q Analysis** plugin. Statistics are generated and data are visualised in the form of spatial-temporal maps.

Additionally, there are data pre-processing plugins available such as:

1. DIC[6] plugin (subsection 12.1)
2. Random Walker Segmentation plugin (subsection 12.2)
3. Mask generator (subsection 12.3)
4. Qconf generator (subsection 8.5)
5. Format converter (subsubsection 14.1.1)
6. Omero client - available from *Tools* → *Omero client* menu (subsection 7.1)

A plugin can be launched from either the [*Plugins* → *QuimP*] menu, or from the QuimP bar (opened via [*Plugins* → *QuimP* → *QuimPBar*]). Running a plugin will either require an image and/or a QuimP parameter file (with extension *.paQP* or *.QCONF*) generated by BOA (see section 14 for details). Demanded file format can be pre-selected by *File Format Selection* tick box. This selection influences all QuimP modules, even if started directly from Fiji plugin menu. Plugins must be executed left to right, although steps can be repeated without re-running previous plugins and fluorescence measurements performed by ANA can be skipped entirely.

Once complete, data can be analysed in Excel or MATLAB.

[6] Differential Interference Contrast microscopy

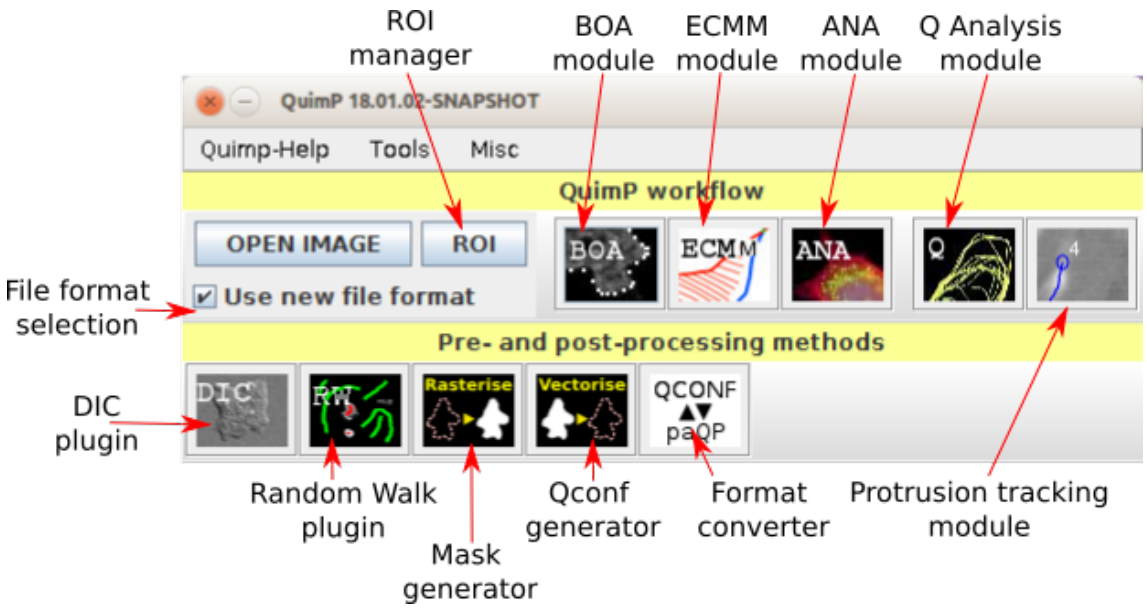


Figure 2: The QuimP bar

7.1 Omero integration

QuimP allows for simple integration with Omero database. Currently supported is uploading and downloading *QCONF* experiment files to and from Omero. In both cases QCONF file and related image are loaded or saved to disk. For more detail consult short help available in OmeroClient module (*Tools* → *Omero client*).

8 Cell Segmentation - BOA Plugin

QuimP utilises an active contour method to segment cells from the background of an image. Typically, this will be a time lapse movie containing a fluorescence channel, captured from a confocal microscope, but the BOA plugin will segment any image which has bright objects on a dark background (note that phase contrast images can not be segmented with this method). Attaining the best image for segmentation may require some pre-processing, such as the combining of available fluorescence channels (the ideal case is that of an evenly white cell on a black background). Figure 3 describes the active contour method and provides insight as to BOA's parameters.

It is expected that objects being segmented have intensity higher than background. Active contour method implemented in BOA will not work for other cases.

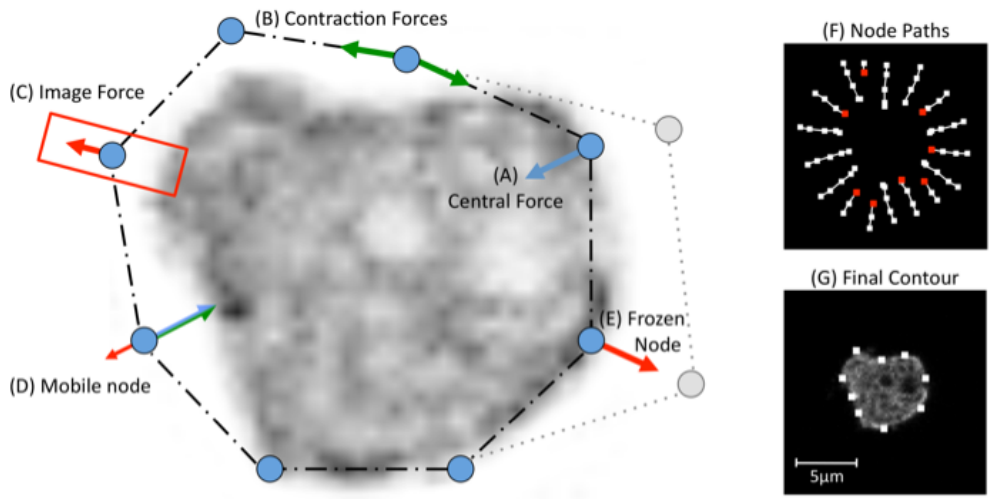


Figure 3: Caricature of outline detection using active contours. QuimP's active contour consists of a chain of connected nodes (a reduced number are shown here for simplicity) initialised to encircle a cell of interest. Three types of forces act on each node; (A) Central forces contribute to chain shrinking; (B) Contraction forces between neighbouring nodes shrink the chain and maintain chain integrity; (C) Image forces oppose the shrinking of the chain by a magnitude determined by the local image intensity gradient (red box). When a node experiences inward forces greater than the opposing image force it moves inwards (D). On reaching the boundary the image force becomes sufficiently large to cancel out the other forces, halting and freezing the node (E). (F) shows the paths of nodes during contraction. Note that nodes can be added or removed (red nodes) to maintain an average distance between neighbours. (G) shows the final position of nodes. QuimP usually makes use of 100 or more nodes to extract high resolution cell outlines. [3]

The BOA window is presented in Figure 4. There are **four** highlighted areas, which contain tools and allow to adjust settings for segmenting and displaying images, and postprocessing of segmented contours.

- Red: segmentation parameters, described in step 4 below and buttons:
  - *Load* - load `QCONF` or *paQP* experiment files. The same as *File→Load experiment*, see subsection 8.1.

- *Save* - update loaded experiment. The same as *File→Save experiment*, see subsection 8.1.
- *Copy prev* - copy segmentation settings from previous frame to current one and re-run segmentation for current frame.
- *Default* - set default segmentation parameters for current frame..

- Blue: controls for navigating through frames in a time series and for zooming cells visible in the current frame.
  - Enabling the tick box labelled *Show paths* will display a trace of the snake as it contracts. Nodes colour pixels white as they move towards the cell boundary. This view can be useful for diagnosing why a segmentation fails.
  - Enabling the tick box labelled *Zoom cell*, will zoom the image to selected cell, and track the zoom to the cell's movements.

- Green: tools for manual editing of contours.
  - *Set Scale* - set pixel size and frame rate. These values can be also read from image file if available or set on the BOA start.
  - *Truncate seq* - allow to truncate segmented sequence.
  - *Add cell* - segment current ROI. Segmented cell should be included entirely in ROI.
  - *Delete cell* - delete already segmented cell. This option removes the whole track of the cell across frames.
  - *Freeze* - exclude cell from further modifications. Any changes of segmentation parameters, plugins or segmentation of the whole sequence will not affect frozen cells.

- Yellow: access to cell contour postprocessing filters (described in subsection 8.3). There are also two buttons here helping in populating current filter stack among other frames:
  - *Populate fwd* - copy current filter stack (together with configuration of used filters) to all next frames (in contrary to menu *Plugin→Populate to all* that copy current configuration to all frames).
  - *Copy prev* - copy configuration from previous frame to the current one.

8.1 BOA menu

- **File**
  - *Load experiment* - load `QCONF` configuration file and restore saved state. One can continue work from point where it was saved in. One can load also old `paQP` files here but program state will not be restored and many configuration options will be missing.
  - *Save experiment* - update already loaded `QCONF` configuration file or create new one if such file does not exist.
  - *Save experiment as..* - save `QCONF` configuration file under specified name.
  - *Load plugin preferences* - load configuration of outline plugins stack.

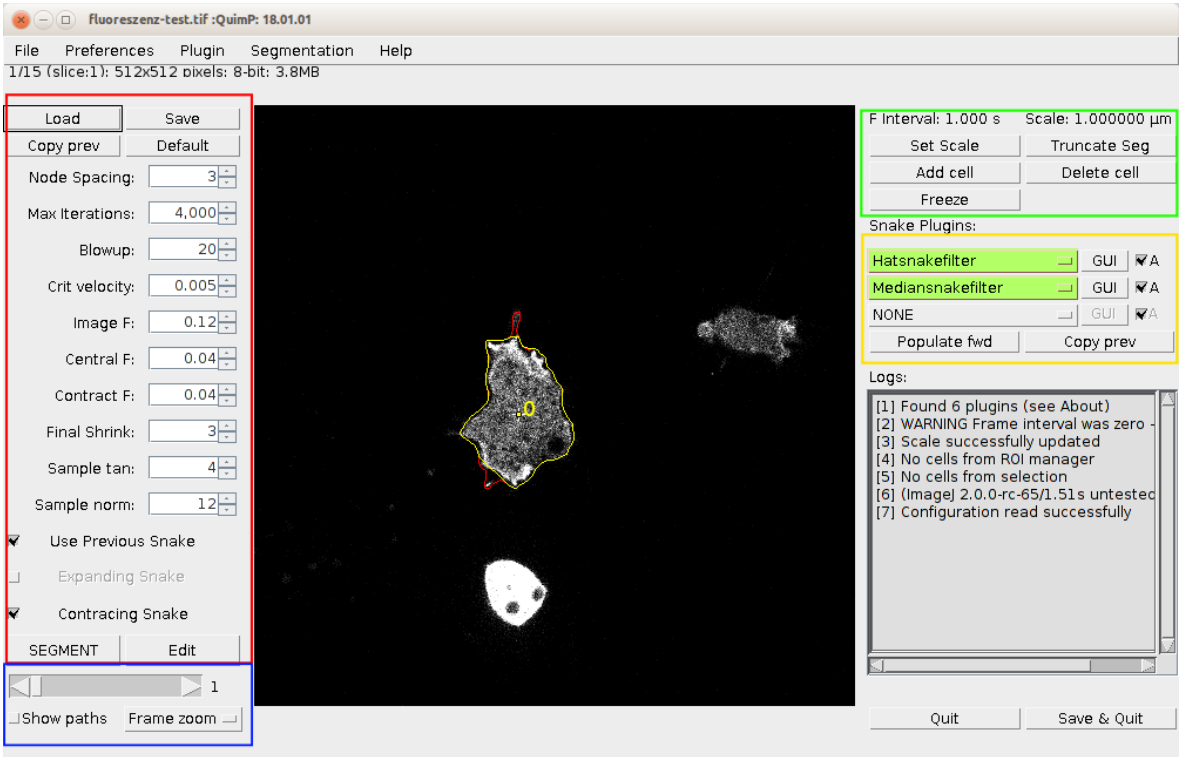


Figure 4: The BOA Window

- *Save global preferences* - save configuration of outline plugins stack.

- **Preferences**
  - *Plot original* - switch on displaying original outlines together with those processed by plugins. If there is no plugin active (or selected) it does nothing.
  - *Plot head* - flag zero node of outline with arrow pointing to the outline direction.
  - *Zoom freezes* - if set, zoom will freeze all cells except that zoomed in. Un-zoom will release all cells.
  - *Show history* - open history window.

- **Plugin**
  - *Discard all* - remove all plugins from all frames. Discard all changes introduced by them to the outline.
  - *Re-apply all* - try to reload plugins from disk and apply them to outlines.
  - *Populate to all frames* - copy plugin tree from current frame to all other frames and apply it to outlines. Do not run segmentation. This option can be useful together with Binary mask segmentation.

- **Segmentation**
  - *Binary segmentation* - perform binary segmentation on black-white mask
  - *Clear all* - remove all segmentation results and set all parameters to their default values

- **Help**
  - *Help contents* - open system web browser and try to load this manual. Internet connection is required.
  - *About* - display version of QuimP and found plugins

8.2 BOA segmentation workflow

Step 1: Open an image and ensure the correct scale. QuimP can segment from a single image, or from an image stack (but not from a hyperstack). Only 8-bit images can be used for segmentation and so you will be prompted to convert to 8-bit if required. QuimP uses the scale properties of the image to scale all output to microns (pixel size) and seconds (frame interval). **ENSURE YOUR SCALES ARE CORRECT** within [*Image* → *Properties*...]. On launching, BOA will prompt to check your image scale. The scale is recorded by BOA and stored in the parameter file for use in the rest of the analysis.

Step 2: Launch the BOA plugin

Unlike previous versions, BOA no longer requires selection of cells prior to launching, but you can do so. Launch BOA from the QuimP bar or menu. The BOA window is now a fully functional ImageJ window that allows the user to perform any ImageJ function, such as drawing, creating ROI's, or zooming while the BOA plugin is in use. Figure 4 shows the BOA window. The image scale is displayed at the top right. To adjust the scale click *Set Scale*.

BOA can segment one, or more cells, each beginning and ending at any frame you wish. In addition, you can manually edit any segmentation at any frame, add/delete cells, truncate segmentations, and adjust the segmentation parameters for individual frames.

Step 3: Adding and deleting cells.

To add a cell first scroll to the frame that you want to begin the segmentation at (either with the mouse wheel, or slider). Using either the circular, rectangular or polygon selection tool, draw an ROI (region of interest) around a cell, and click *Add cell*. BOA will immediately attempt to segment the cell at the current frame. If convergence is successful, the result is displayed as an image overlay. The cell is also given an identification number. If the segmentation fails for some reason, the parameters can be adjusted (see below). You do not need to re-add the cell.

Additional cells can be added in the same way. The active contours surrounding each cell will interact with one another and prevent contours crossing over to other cells. This capability is particularly useful in situations where cells come into close contact, which often disrupts the segmentation process.

If you are not able to find optimal parameters for many cells at once, you can select only one cell then follow points below and at the end use *Freeze* button to block further modification of this cell. Then repeat the whole process for another cell. Note that *Freeze* freezes the whole sequence and one can has many sequences frozen at one time.

To delete a cell click *Delete cell*, and then click the centre marker of a cell, on any frame where a segmentation is visible. Hit the same button again to leave delete mode.

Step 4: Adjusting segmentation parameters.. When a parameter is altered, BOA immediately re-computes the segmentation for all cells on the current frame with the new parameters (all cells share the same parameters). Each has a minimum/maximum value that can not be exceeded. To alter a parameter either enter a value into the text box, or use the arrows to increase/decrease in steps. The key to attaining a good segmentation is the balancing of the 3 forces. First, try adjusting the *Image F* parameter to improve the result. Parameters can be set for each frame separately. One can also freeze cell to exclude it from further processing.

The parameters are as follows:

- **Node spacing** - Roughly equates to the number of pixels between nodes, and hence the resolution of the segmentation.
- **Max iterations** - The maximum number of iterations applied to shrink the contour.
- **Blowup** - The number of pixels to expand the solution to the previous frame, for beginning calculation at the current frame (utilised when 'Use previous snake' is active). Increase if your target cell is moving by large amounts between frames. Decrease if other cells are coming into close proximity and interfering with the snake. Lower values reduce computation time.
- **Critical velocity** - The cut-off speed at which nodes are frozen. Lower the value to allow the snake to continue to contract for longer, which may aid in letting nodes enter concavities.
- **Image F** - Controls the magnitude of the image force pushing nodes outwards. This is the most effective parameter and should be the first port of call for adjustment. Highly intense cells will produce high image forces, and you may need to lower this value to allow nodes to approach the cell outline. With weakly intense cells the snake may collapse inwards, past the cell boundary, in which case increase the image force.
- **Central F** - The magnitude of the central force pulling nodes inwards. Increase to shrink the snake further and help nodes enter concavities. Decrease to prevent nodes entering the interior of cells.
- **Contract F** - Magnitude of the force pulling nodes together (i.e. contracting the snake). Increase to prevent nodes 'falling through holes', or to produce a smoother final solution.
- **Final shrink** - The number of pixel to shrink the solution to tighten the snake to the cell boundary. A value of zero does not shrink the solution at all.
- **Sample tan, Sample norm** - Defines (in pixels) the size of the sampling box around each node within which the image is sampled for calculation of the image force. 'tan' describes its width, 'norm' its depth. A larger sampling box will increase the size of the image force. Briefly, the sampling box should be an appropriate size as to give an accurate sample of the local intensity between the cell and the background.
- **Use previous snake** - If selected, the solution to the previous frame will be used as the starting contour for the current frame. Otherwise, the users selection is used as the starting contour for every frame.
- **Expanding snake** - Experimental option. Rather than contract the snake, it expands it. Initial contour must be drawn inside the cell. Note that all forces are reversed now.

Buttons and checkboxes:

Previous segmentation for your image can be loaded back into BOA by hitting load and selecting a QuimP parameter file with the file extension either *.paQP* or *.QCONF*. Parameter values alone can be loaded by declining to load associated snakes. Button *Default* sets above parameters to their default values and recompute current frame. The *Copy prev* button will copy segmentation parameters from previous frame to the current one and re-run segmentation.

Enabling the tick box labelled *Show paths* will display a trace of the snake as it contracts. Nodes colour pixels white as they move towards the cell boundary. This view can be useful for diagnosing why a segmentation fails.

Step 5: Segmenting multiple frames. When happy with the segmentation on the first frame, hit *SEGMENT* to segment the cell (or cells) in the following frames. Segmentation parameters from starting frame will be copied to subsequent frames. Any outlines already existing on these frames will be overwritten. The algorithm may fail at a particular frame and a warning will be printed to the log window, at which point you can alter parameters.plugins. This process can be stopped in any time hitting *SEGMENT* once more.

Step 6: Correcting segmentations. In the case of failure, or incorrect results, you have 3 options:

- **1 - Edit a solution.** Scroll to a frame where an error occurred, click *EDIT* (If you have more than one cell click its centre). Drag the points of the ROI to the correct locations (you can hold *alt* to delete nodes or *shift to split nodes*). You can zoom using the imageJ tool (magnifying glass). If you loose your ROI go to [*Edit* → *selection* → *restoreselection*]. Scroll to edit other frames. When finished, leave edit mode by clicking **STOP EDIT**.
- **2 - Adjust the parameters.** Scroll to a frame where an error occurred and change the parameters to improve the segmentation. Optionally, click *SEGMENT* to apply these new parameters to the following frames.
- **3 - Truncate segmentations.** Scroll to the first frame where an error occurred and click *Truncate Seg'*. Click the centre of a cell to remove all segmentations from the current frame onwards.
- **4 - Use BOA plugins.** Try to apply available BOA filters for particular frames or for whole stack. Check *Plugin* menu (subsection 8.1) and subsection 8.3 for details.

You can use also *Zoom freezes* feature from *Preferences* to change *Frame Zoom* behaviour. If *Zoom freezes* is ticked on, *Frame Zoom* freezes all already segmented cells except that currently zoomed. Then any modification will apply only to magnified cell.

Step 7: Save the results. Click *Save and Quit*. The saved cell outlines are those visible on the yellow overlay. For new *QCONF* format selected (default) only one composite file is outputted. Otherwise a set of files will be saved for each segmented cell (see section 14 for description differences between new and old data file format). A cell's ID number is automatically added to filenames.

Enter a name for the analysis (file name extensions are added automatically). BOA outputs files with a *QCONF* extension for new fileformat or *.xxQP* for old one[7].

[7] QuimP11

Files extended with *.paQP* contain file paths and parameters associated with a particular analysis. When running other QuimP plugins, it is the *.paQP* file which you must select to continue an analysis (see subsection 14.2).

Files extended *.snQP* contain all data associated with individual nodes of an outline, including pixel coordinates, local cortex intensity, local membrane velocities, and tracking data (see subsection 14.3).

Files extended *.stQP.csv* contain average statistics per frame, and can be opened directly in Excel as a 'comma separated file' (see subsection 14.4).

The BOA plugin alone outputs many useful statistics regarding cell morphology and migration, without the need to run any further analysis. Simply open the *.stQP.csv* file in Excel, or use the accompanying scripts to load data into MATLAB (see section section 16).

8.3 Filtering the outline

BOA module supports external plugins for post-processing outlines on current frame. Plugins are standalone *jar* files that can be developed by other scientists and distributed independently from QuimP. Available plugins are discovered when BOA launches by scanning default location, which is Fiji/plugins folder. There is no need of any additional action, discovered plugins are available in filter stack on right (see Figure 4). There are three slots in the stack. Filters are applied in order from most top to bottom, next filter starts with contours returned from the previous one. Any action on filters relates only to current frame (thus each frame can have different filter stack) unless *SEGMENTATION* button is clicked. Automatic segmentation propagates most recent configuration (that means all options available in BOA window) from starting frame to all subsequent frames. Modified contour is drawn in standard yellow colour, whereas red denotes original unprocessed result of segmentation (Figure 5). Any further processing in QuimP workflow (e.g. ECMM or ANA) applies to filtered outline. Refer to subsection 8.1 for other options related to filters. *Frozen* cells (section 8) are also excluded from processing by filters.

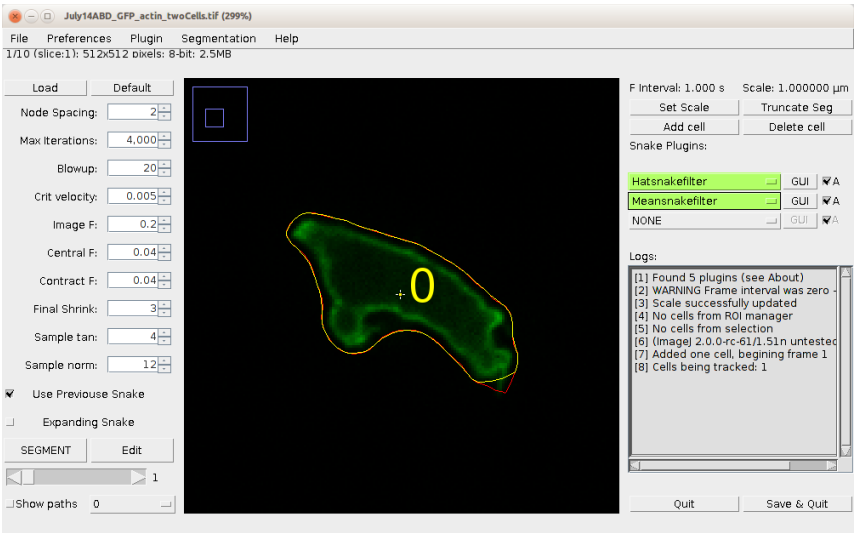


Figure 5: BOA filters in action.

If filter contains an user interface, it can be accessed by *GUI* button (Figure 5).

Default installation of QuimP contains four BOA filters:

- **HatSnakeFilter** Removes small protrusions from outlines.
- **MeanSnakeFilter** Apply running mean filter to outlines.
- **MedianSnakeFilter** Apply running median filter to outlines.
- **SetHeadSnakeFilter** Allow to choose first node in snake.
- **RandomWalkSnakeFilter** Perform Random Walk segmentation of frame using Active Contour results (outlines) as initial seed. Refer to subsection 12.2 for more details about Random Walk.
- **HedgeHogSnakeFilter** For testing purposes.

Filters that have been discovered and properly loaded are marked in green colour. This is standard behaviour when user start analysis from scratch. If BOA configuration has been restored from *QCONF* file, filters (if any was used) are displayed in red colour to denote that result of segmentation (that is stored in QCONF file together with raw unprocessed outlines) was processed by filter stack but none of those filters is loaded yet to BOA. In this mode user can browse through frames, check segmentation parameters, etc. even if a filter is physically not available in his configuration. Option *Plugin→Re-apply all* will try to load all necessary filters from disk and apply them to each frame where they were used. Note that this operation will recompute all outlines on frames where filters were used and replace loaded results with those evaluated.

8.4 A Few Words on Image Segmentation

If you run into difficulties attaining a good segmentation, there are a few tricks that may help. If the width of your cell is around 20 pixels or less try artificially increasing the size of the image with some form of pixel interpolation selected ([$Image \rightarrow Adjust \rightarrow Size$]). This should aid sampling of the image and allow nodes to enter concavities with greater ease (note, all subsequent analysis must be performed on the image at the new resolution).

If your images are particular noisy, and cell's appear very grainy, you may try applying a weak Gaussian blur to smooth them ([$Process \rightarrow Filters \rightarrow GaussianBlur$]). This should make for more reliable sampling of the cell's interior.

Remember to recalculate the scale of the image when resizing, and to perform intensity analysis on the rescaled image without any interpolation.

Another common approach to improving image quality is to remove background noise using a 'blank' image, one taken without any cells in the field. This is particularly useful for removing shadows of dirt on the optical equipment, and correcting uneven backgrounds.

8.5 Binary mask segmentation

The QuimP supports creation of Snake outlines directly from binary masks. As the active contour method is not used in this case, one can use another segmentation algorithms (e.g. Random Walker method) to deal with problems that typically occur for active contour such as e.g. bad segmentation of cavities.

To use binary segmentation, either black-white mask (8-bit image in ImageJ) or grayscale images should be obtained in any other way, e.g. by using alternative segmentation tools. It has to be size of segmented image that is already loaded to BOA plugin. The number of slices in mask file should be less or equal to the number of slices present in segmented image. **The background pixels must have value 0 whereas objects can be defined by any other values within the range 1-255**. This plugin is capable to perform simple cell tracking between adjacent frames. If input mask is binary then cells are assigned to the same track by testing their overlapping between frames. For grayscale images, numeric values of pixel intensities are used.

The binary segmentation plugin can be launched from BOA menu (*Segmentation* → *Binary segmentation*) or as standalone module from QuimP Toolbar. The latter method supports also IJ macros. If one calls this plugin as separate external module from QuimP Toolbar or IJ menu, it will produce Qconf file in the same way as BOA module does. This file can be than passed to other QuimP modules for further analysis.

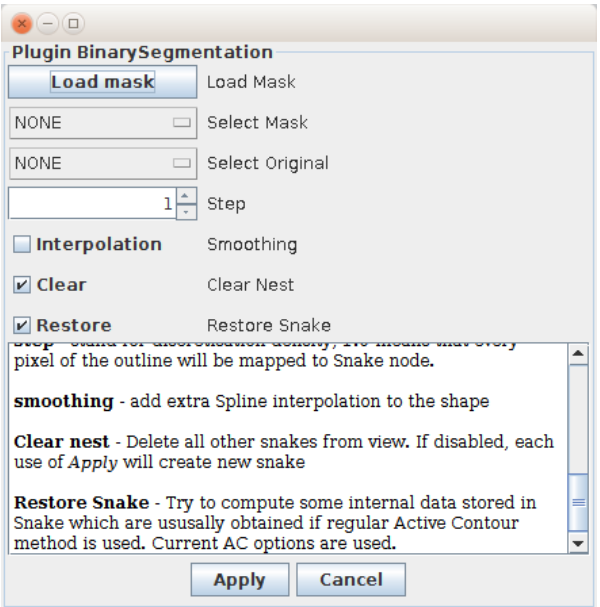


Figure 6: The Binary Segmentation Window

The mask can be either loaded from separate file (ImageJ formats are supported) or selected among already opened images in ImageJ. Results of segmentation are displayed in real-time on BOA window immediately after clicking *Apply*. *Cancel* closes the plugin window leaving last results of segmentation in BOA. If *Clear* is selected each use of *Apply* will clear current outlines in BOA. If it is unselected, outline will be added to BOA. Note that each click on *Apply* will create new outline even if non of plugin parameters is modified. Resulting outline can be smoothed by e.g. `MeanSnakefilter` available in the plugin area (see Figure 4). Filter can be applied for selected frames only or populated for the whole sequence by (*Plugin* → *Populate plugins*)

9 Cell Tracking - ECMM Plugin

To extract data on local membrane velocity QuimP utilises our Electrostatic Contour Migration Method (ECMM). Nodes that form a cell outline at time t are mapped onto the segmented outline at time *t+1*. The distance a node migrates during the mapping process, and the frame interval, determines the local membrane velocity at the node's position. If you only have a single frame there is no need to run this plugin. ECMM edits and adds to data already contained within an *.snQP* file.

Step 1: Launch the ECMM Mapping plugin, and open the desired *.paQP* file when prompted. There are no parameters that require manual setting.

Step 2: Inspect the results. The blue outline represents the cell outline at time t, the green at time $t + 1$. Green circles, overlaid with crosses, denote intersection points that have been validated and used for calculating sectors. Red lines denote paths taken by migrated nodes. Nodes that fail to map correctly are highlighted in yellow. Failed node (FN) warnings are displayed along with a running total (in brackets) of the number of failed nodes in the whole sequence. The log window will display progress and warnings.

If a node fails to map it is simply ignored, the effect of this is a small reduction in mapping resolution at the nodes location. If the process fails to map the majority of the nodes correctly, then a new segmentation will have to be performed (try using a different node resolution or changing the image resolution).

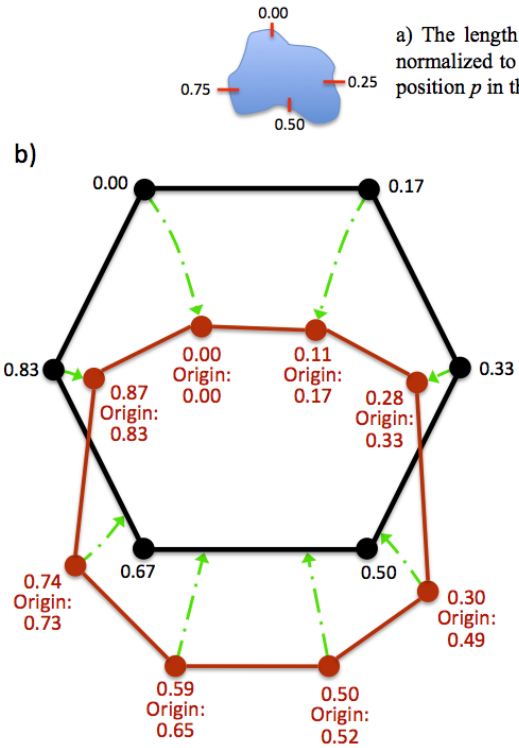
9.1 ECMM Tracking

The details as to the workings of ECMM can be found in our publication [3]. The current implementation includes several significant improvements. It is not necessary to understand the method, only the format of the tracking.

At frame $t = 1$ a node is randomly chosen as being node zero, n_0. Other nodes are then assigned a position according to their distance from n_0 along the cell perimeter. The length of the cell perimeter is normalised to 1, hence the position of a node which is exactly half way around the perimeter will be 0.5 (see Figure 7a).

The nodes of the outline at $t = 2$ also have positions assigned, but in addition have an *origin*. This represents where a node originated from on the $t = 1$ outline. For example, a node with an origin of 0.5 originated from position 0.5, exactly half way around the outline at $t = 1$.

It should be realised that this method of tracking is sub-node resolution. By simply interpolating the positions and origins of nodes, any real number position on an outline can be tracked at sub-node resolution, for as many frames as desired. See subsection 14.3 for the format of data output, and subsection 16.3 for using ECMM data within MATLAB.



c) Resulting Data

Position	Origin	Speed
#frame 1		
6		
0.00	-	-
0.17	-	-
0.33	-	-
0.50	-	-
0.67	-	-
0.83	-	-
#frame 2		
8		
0.00	0.00	-6.1
0.11	0.17	-6.2
0.28	0.33	-1.1
0.30	0.49	+3.1
0.50	0.52	+5.4
0.59	0.65	+5.3
0.74	0.73	+2.9
0.87	0.83	-0.9

Figure 7: **ECMM tracking method.** **a)** Normalisation of the cell perimeter. **b)** Caricature of the tracking of node positions between two frames (black t, red $t+1$). Arrows pointing towards the red outline indicate forward mappings and a nodes origin is recorded simply as its position on the black outline. Arrows pointing towards the black outline indicate reverse mapping and node origins are recorded as interpolated values between nodes on the back outline. **c)** Data from *b)* as it would appear in a *.snQP* file. The frame number is followed by the number of nodes in the outline. Speeds are proportional to the lengths of the green mapping arrows.

10 Fluorescence Measurements - ANA Plugin

QuimP samples the intensity of pixels from within the cortical region of a cell. At each frame, nodes of the cell outline are shrunk inwards to form an inner outline. The amount of shrinkage is specified by the user, and relates to the estimated cortex width. Nodes are migrated inwards towards the inner outline, while simultaneously measuring pixel intensities (a 3x3 average). A nodes fluorescence intensity is the maximum recorded as it migrates across the cortex. ANA can record data for up to 3 separate fluorescence channels.

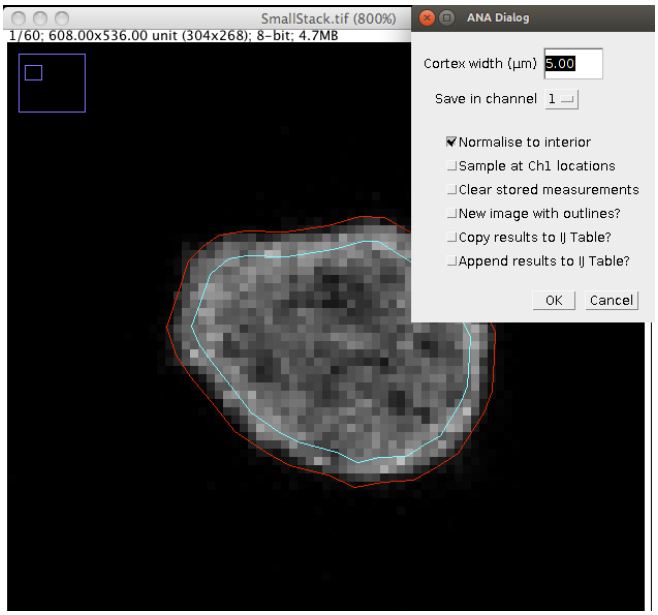


Figure 8: ANA. The area between the red and blue contours is the user defined cortex, and is where intensity samples are taken. If *Normalise to interior* is ticked, sampled intensities will be normalised to the average intensity of pixels within the blue contour.

Step 1: Open an image stack containing the desired fluorescence channel. This may be the same stack used for cell segmentation, or one containing data from another channel. Ensure the opened stack has the same resolution and number of frames as that used for segmentation.

Step 2: Launch the ANA plugin and open the desired *.paQP* or (recommended) *QCONF* file when prompted.

Step 3: Enter a value for the cortex width (μm) and choose a channel. The inner and outer outlines are displayed on the image. Changing the cortex width will update the image overlay. Choose a channel to record data in from the drop down menu.

ANA provides 5 further options

- Normalise to interior - Toggle normalisation of intensity sampling to the cells interior (recommended).
- Sample at Ch1 locations - Sample at the same locations as determined for channel 1.
- Clear stored measurements - Removal all fluorescence measurements from QuimP's files.

- New image with outlines? - Produce copy of input image stack with inner and outer outlines in it. Outlines are drawn at overlay[8]. Flatten the image before using it in another software.
- Copy results to IJ table? - Open standard ImageJ table[9] filled with fluorescence statistics computed for selected channel (the same are written to*stQP.csv* file). Note that is an experiment contains more than one cell, ANA will be run for each of them separately but finally the table will always contain statistics for all cells.
- Append results to IJ table - works as *Copy results to IJ table?* but does not clear table before. Can be used for collecting results from different experiments.

Click *OK*.

Fluorescence data associated with specific nodes is saved in the *QCONF* file and additionally in *.stQP.csv*.

Step 4: For further channels, open the next channel image and re-run ANA.

[8] https://imagej.nih.gov/ij/docs/guide/146-11.html
[9] Standard table always is named *Results*

11 Compiling Data - Q Analysis Plugin

Data within *QCONF* file (or *.snQP* and *.stQP* for old path[10]) files can be read into Matlab at this point, but motility maps have not yet been generated[11]. The Q Analysis plugin will build spatial-temporal maps of motility, fluorescence, and convexity, and also several other maps to aid analysis. In addition, vector graphics are produced using the *Scalable Vector Graphics* format. These include a cell track in which all cell outlines are overlaid and coloured according to frame number, and a motility movie in which nodes are coloured with a user specified data type.

Step 1: Launch the Q Analysis plugin and open the desired *.paQP* file when prompted.

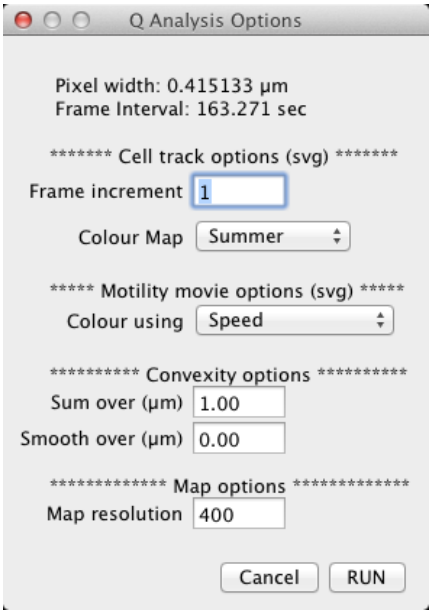


Figure 9: Q Analysis options dialog

Step 2: Enter parameters to the options dialog. The parameters are as follows:

- **Cell track option - Frame increment** [positive integer]. The frame increment for drawing the cell track (*_track.svg*). Frames are only draw at the specified increment. Set to 1 for all frames.
- **Cell track option - Colour Map**. Select a colour scheme for the cell track.
- **Motility movie option - Colour using.**. Select the data type for use in colouring nodes of the motility movie.
- **Convexity option - Sum over (microns)** [positive real number]. Distance around the cell perimeter (μm) to sum curvature. Set to zero for no summation.
- **Convexity option - Smooth over (microns)** [positive real number]. Distance around the cell perimeter (μm) to over which to average curvature for smoothing. Set to zero for no smoothing.

[10]Refer to section 14

[11]If old path is selected, otherwise one should use Format Converter tool to obtain these files. Refer to section 14

- **Map option - Map resolution** [positive integer]. A value of x will produce images that are X pixels in width. The height of the image will be equal to the number of frames of the analysed sequence, or a multiple of number of frames (if less then x). Data is interpolated to form maps of any size.

The images produced are for visualisation only, as they have been scaled to fill the RGB colour spectrum. Vertical image scale is set to frames (frame zero at the top), horizontal to the normalised length of the cell perimeter (0,1]. Import *.maQP* files as text images for further analysis in ImageJ. See Section subsection 14.5 for details and analysis of maps.

12 QuimP preprocessing plugins

12.1 DIC images processing

Differential interference contrast microscopy (DIC) enhances the contrast in unstained, transparent samples. DIC images can be easily interpreted and analyzed by biologists due to high resolution available and excellent contrast generated by steep gradients in optical path lengths and phase shifts between the two polarized beams. Nevertheless, very characteristic bas-relief observed in DIC images is the source of problems in automatic analysis of these images. The image contrast and the same the objects boundaries are well defined in the shear angle but in direction perpendicular to it there is no contrast against to background and thus a lack of distinct edges of object. Moreover, strong gradient of image intensity at shear angle negatively influence standard image processing methods like global thresholding or edge detection producing insufficient results, with discontinuous regions or edges.

The DIC plugin provided with QuimP allows to convert DIC samples into form more suitable for segmentation and further image processing. The local contract for any cell in reconstructed image is well defined in every direction and the intensities of pixels take only positive values (in relation to a mean intensity level). The algorithm implemented in QuimP is described in [2]. Either single images or time lapse movies from DIC microscopy can be processed.

Step 1: Open an image. Once loaded into ImageJ/Fiji click DIC filter in QuimP bar. Only 8-bits images are supported (256 gray levels).

Step 2: Set correct parameters There are two important parameters: *Shear* angle and *Decay* (Fig. Figure 10). The *Shear* parameter is the shear angle of DIC microscopy measured counterclockwise, whereas *Decay* is described in details at [2]. The *Decay* factor depends mainly on the size of cells and the best results are usually achieved experimentally. *Invert output* option inverses colorscale in reconstructed image.

Reconstructed image can be affected by linear pattern that can be filtered by running mean filter activated if *Filter mask size* is greater than 0. Smoothing filter is applied before processing for angle perpendicular to given *Shear*[12].

If objects (cells) in reconstructed image are darker than background, the image should be inverted before further processing in BOA module. One can use ImageJ option `Edit→Invert` or tick *Invert output* in plugin window. Refer to section 8.

12.2 Random Walker Segmentation

The Random Walks Segmentation algorithm is described in the paper [1]. It belongs to the class of supervised segmentation algorithms what means that the user interaction is needed. In RW method a user has to specify a small set of pixels belonging to the desired object and (possibly) a set of pixels belonging to the background. Those pixels are called *seed pixels* and in this plugin they form the

[12] Currently, smoothing can be used only with the following shear angles: 0, 45, 90, 135. For another values it will run for closest multiplicity of 45 degrees.

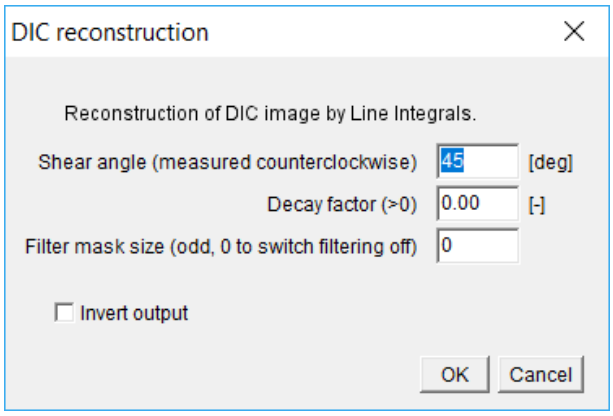


Figure 10: DIC filter options dialog

seed image. The seed image must be the size of segmented image whereas, the number of layers can be either equal to number of layers in segmented image or set to 1.

This plugin supports also multi-object segmentation, where user defines several foreground objects (up to 255). For this case, the output is a grayscale image where each unique object is marked by different colour, starting from 1.

Segmentation is run in two sweeps. First one gives rough segmentation whereas the second one produces final result. Second sweep uses binary mask produced by the first step as an input after filtering it by selected *Binary filter*. Second sweep can be disabled by setting *Gamma 1* to 0 (disabled by default, use with care as it can give unpredictable results in some cases). By default second sweep uses half of defined iteration number.

For best results, the background seeds should be drawn close to foreground objects.

Implemented algorithm uses iterative scheme with three possible stopping criteria:

1. ITERATIONS - if number of iteration defined in *Iterations* is exceeded for first sweep or half of this number for the second sweep (if enabled).
2. NANS - if Infs or NaNs appear in solution (they will be removed before outputting result)
3. RELERR - if relative error is smaller than defined. Defined by *Rel error*.

The RW plugin can be called outside of QuimP from the QuimP Bar, from IJ macro or directly from API. The main interface is presented in Figure 11. The window is divided into eight sections:

1. Image section
2. Seeds section
3. Dynamic section that depends on selection in previous section
4. Segmentation options section
5. Inter-processing section

6. Local mean feature section
7. Post-processing section
8. Display options section

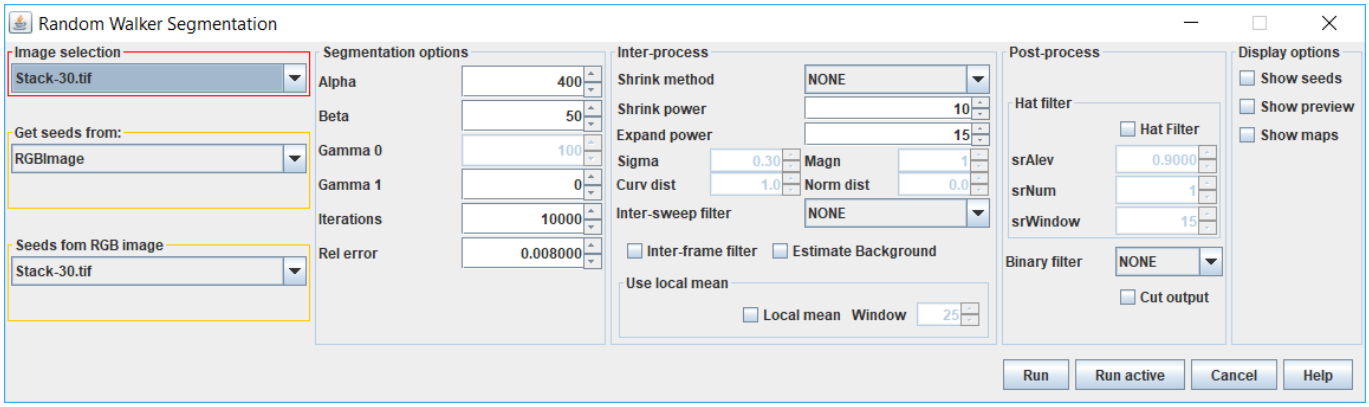


Figure 11: Random Walker segmentation dialog

Image section - here user can select image that will be segmented. It must be grayscale image either 8 or 16 bits.

Seeds section - defines source of the seed image. The following options are available:

- RGB image - user can provide RGB image where foreground pixels are labeled with pure red colour and background with pure green. The z-dimension of the seed image must equal to one or to the number of slices of segmented image. If there is only one slice of the seed given, it will be applied to the first slice of segmented image. For the n+1 slice the seed will be generated using results of segmentation of n-th slice processed by algorithm selected in *Inter-processing section*. In this mode only one foreground object is supported.
- Create image - user can create RGB image cloned from the image selected in *Image section* (*Clone* button). The *FG* and *BG* buttons select foreground and background pen respectively.
- Mask image - user can provide grayscale image, where pixels of values grater than 0 stand for the foreground seeds. This mode is intended for using results of Active Contour segmentation as input to Random Walk algorithm. The seed stack should have the same size as segmented image. The foreground labels can be larger than objects they label because they are pre-processed (usually shrank) by algorithm selected in *Inter-processing section*. Typical use case is to perform rough segmentation in BOA module using fail safe parameters which usually lead to rough segmentation and then to use Random Walk plugin to get fine one. The improvements should clearly visible especially in concave areas where Active Contour method fails. **Note** - providing mask as initial seed usually requires any *Shrink method* to be set.
- QCONF file - the same as *Mask image* but user can provide *QCONF* file saved in BOA module. Cells stored in the file are rendered to grayscale image, each cell is segmented independently from others.

- ROIs - initial seeds can be provided as ROIs. This mode also supports multi-object segmentation. There is simple tool available for collecting ROIs, described in subsubsection 12.2.1.

Dynamic section - its content depends on selected seed source.

Segmentation options - controls segmentation process. Options available here:

- *Alpha* - penalises pixels whose intensities are far away from the mean seed intensity
- *Beta* - penalises pixels located at an edge, i.e. where there is a large gradient in intensity. Diffusion will be reduced.
- *Gamma 0* - currently disabled
- *Gamma 1* - set to 0 to skip second sweep. Other values are currently ignored.
- *Iterations* - number of iterations (half of this number for second sweep)
- *Rell error* - Relative error - stopping criterion for iterative Random Walk algorithm

Inter-processing section - it controls how binary masks should be transformed to seeds internally. This process applies when seeds are given as result**Seeds section**s of segmentation from other method (*Mask image* or *QCONF file*) or there is only one slice of seeds provided (*RGB image*). Binary mask is shrunk by algorithm selected in **Shrink method** with **Shrink power**. Results define object's seeds for next frame. Then, the same mask (unmodified) is expanded by the same algorithm but with **Expand power** and then inverted to define background seeds for the next frame. Options available here are:

- Shrink method:
  - *NONE* - no shrinking or expanding. Mask is used directly as seed
  - *CONTOUR* - the ANA contour processing algorithm is used for modifying masks. They are first outlined then vectorised and finally processed. This approach preserves shape of the contour. Parameters *Shrink power* and *Expand power* relate to number of pixels to shrink and expand the contour respectively. Shrink power can be additionally function of local curvature, which is controlled by the following options:
    * *sigma* - sigma of Gaussian function used for computing *Shrink power* multiplier for negative curvature. For positive curvature multiplier is explicitly set to 1.0.
    * *Magn* - maximal value of multiplier. The Gaussian curve is scaled to give value of 1.0 around curvature of 0 deg and *Magn* for large negative curvatures. Its slope depends on *sigma*. Set it to 1.0 to get curvature independent scaling.
    * *Curv dist* - approximate number of nodes used for local averaging of curvature along outline. Values less than 3 usually give average over three nodes (current one, next and previous).
    * *Norm dist* - approximate number of nodes used for normales equalization. The algorithm looks for node with minimal curvature within this range and then set normales for all nodes in processed window to be same as in this node. Refer to Figure 12 for examples of *Curv dist* and *Norm dist* parameters.

Generally, non-linear shrinking try to maintain shape of original contour but constricting it more around concave regions of the outline, that should reduce accidental miss-labeling the background.

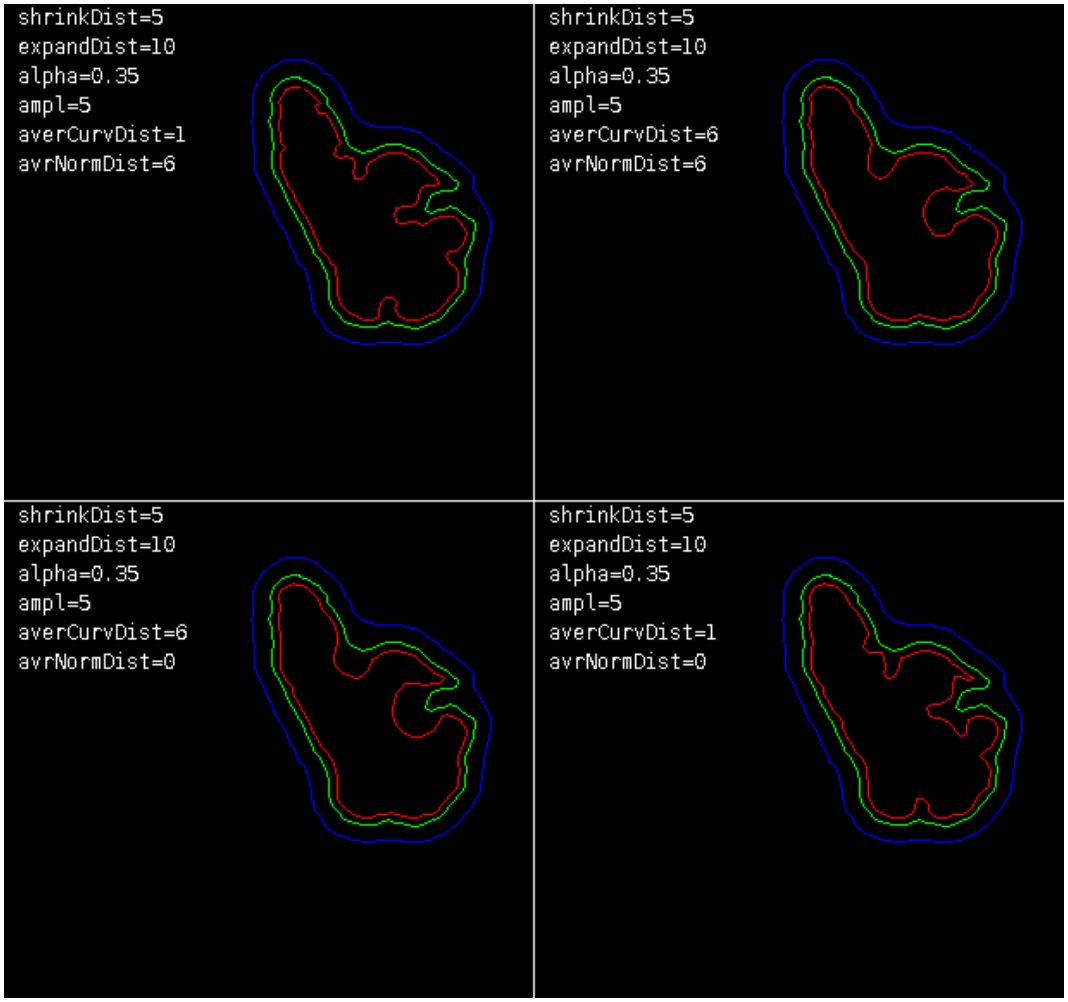


Figure 12: Nonlinear contour shrinking. Examples of different settings. Green - original contour, red - constricted contour, blue - expanded contour.

  - *MORPHOLOGICAL* - masks are processed by morphological erosion. Parameters *Shrink power* and *Expand power* relate to number of algorithm iterations.

- Inter-sweep filter - applied between sweeps. It is intended to remove binary noise, isolated pixels, etc, that could give false seeding:
  - *NONE* - filtering disabled
  - *SIMPLE* - performs one iteration of morphological opening operation.
  - *MEDIAN* - median filtering with kernel of radius 2.
- *Estimate Background* - try to apply simple Otsu thresholding to background seeds. Use *Show seeds* to verify correctness of estimation.
- *Inter-frame filter* - apply additional erosion and median filtering between frames during propagation of seeds.

Local mean - this option can improve behavior of the algorithm in areas with high gradient of intensity, usually at cell cortex, preserving cell boundaries. This method works only with seeds provided as *Mask image* or *QCONF file*. Moreover, masks should be larger than the object itself. *Window* parameter stands for the size (odd) of the sampling window used for computing local mean intensity.

Post-processing section - it covers filtering of results applied after segmentation. The options are:

- Hat filter - it tries to remove small inclusions in contour that typically came from vesicles located near cell edge. The algorithm is described here.
  - *srAlev* - acceptance level beyond the inclusion is removed
  - *srNum* - number of inclusions to remove. All they must have rank above *srAlev*. If this parameter is set to 0, any inclusion with rank above *srAlev* will be removed.
  - *srWindow* - size of the processing window in pixels. Smaller window is more sensitive to small inclusions.
- Binary filter - stands for the final filtering, the same algorithm are used as in *Inter-processing section*.
- Cut output - output mask will be cut by input mask. This option assures that resulting mask is not larger than the seed mask.

Display options section - allows to show real time preview of each segmented frame and seeds computed as results of *Iter-processing* binary masks. This option can be used for verifying *Shrink power* and *Expand power* settings. *Show maps* display raw unscalled probability maps for each seed. This option does not work with stacks, only probability maps for last segmented frame are displayed.

12.2.1 Multi cell segmentation - Collecting ROIs

ROI collector is available after selecting *ROIs* in **Seeds section**. This tool uses ImageJ ROI Manager to store ROIs that denote foreground and background seeds. The ROIs need to obey certain naming criterion, such as *[LB][Id]_[No]*, where *LB* can be either *fg* or *bg* denoting foreground or background seed respectively, *Id* is unique number of foreground object (only one or none background seeds are allowed) and *No* is a sequence number of ROI for object of specified *Id* (a seed for one object can be specified by several separate ROIs).

The ROI collector tool (Figure 13) manages proper naming of ROIs in the ROI Manager. After launching, the ROI Manager will be opened and its content erased.

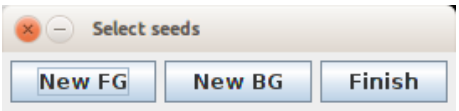


Figure 13: ROI collector tool used for selecting foreground and background objects.

User can label an image using standard ImageJ ROI objects and then click *New FG* or *new BG* to add current selection to ROI Manager. Each next use of *New FG* will add new foreground object.

If certain object ought to be labeled by separated ROIs, they should be added to ROI manager first (by *Add* button) and then *New FG* should be clicked. All ROIS in ROI Manager that do not follow specified naming convention are renamed and assigned to **one** object by *New* * buttons. To finish labeling and transfer ROIs to the plugin click *Finish* button.

12.2.2 Random Walker Segmentation Workflow

If segmentation ends too early (e.g. result is very similar to seeds) one can open Fiji Console (*Window*→*Console*) and check reason of stopping and relative error. If process stops due to reaching relative error in the first few iterations, one can adjust *Rel error* accordingly.

12.3 Mask generator

This plugin can convert cells contours into binary image mask. The cell shape is represented by white area being a filled cell contour whereas the background in black. Masks are widely used in computer graphics for selecting regions of image for processing and analysing. ImageJ is also equipped with relevant procedures that allow for explicit processing masked regions. Therefore, this plugin make it possible to continue analysis of the sample outside the QuimP infrastructure.

Mask Generator plugin works with *QCONF* files saved by BOA module. It is not possible to use it with old file format (paQP) unless it has been converted to *QCONF* by Format Converter (available from QuimP Toolbar menu). By default, the mask image is displayed on the screen and saved on disk at location of input *QCONF* file with suffix *snakemask*.

This plugin supports ImageJ macro language. Use *Macro Recorder* to discover the syntax.

12.3.1 Suggested workflow

- Run Mask Generator plugin from QuimP toolbar (Figure 2).
- Point to *QCONF* file to process. This plugin does not support old *paQP* files (but they can be converted to ne file format by Format Converter, refer to section 14).
- Binary mask image, converted from contours stored in QCONF file by BOA module will be displayed on the screen and saved on disk (Fig. 3). Default saving location is the location of opened QCONF file. The quality of mask depends on the quality of segmentation and the number of nodes configured in BOA module.
- Use `Edit`→`Selection`→`Create Selection` to convert binary mask to ImageJ ROIs. Then, it can be populated through ROI manager to e.g. source image.

13 Protrusion tracking

Protrusion Analysis module allows investigating protrusion formation and dynamics. The module requires *QCONF* file that contains full description of QuimP analysis. Particularly, the BOA, ECMM an Q Analysis modules should be run on *QCONF* before using Protrusion Analysis.

Arbitrary points of interest can be selected directly on the cell contour or imported from *ROI Manager*.

The user interface is shown below.

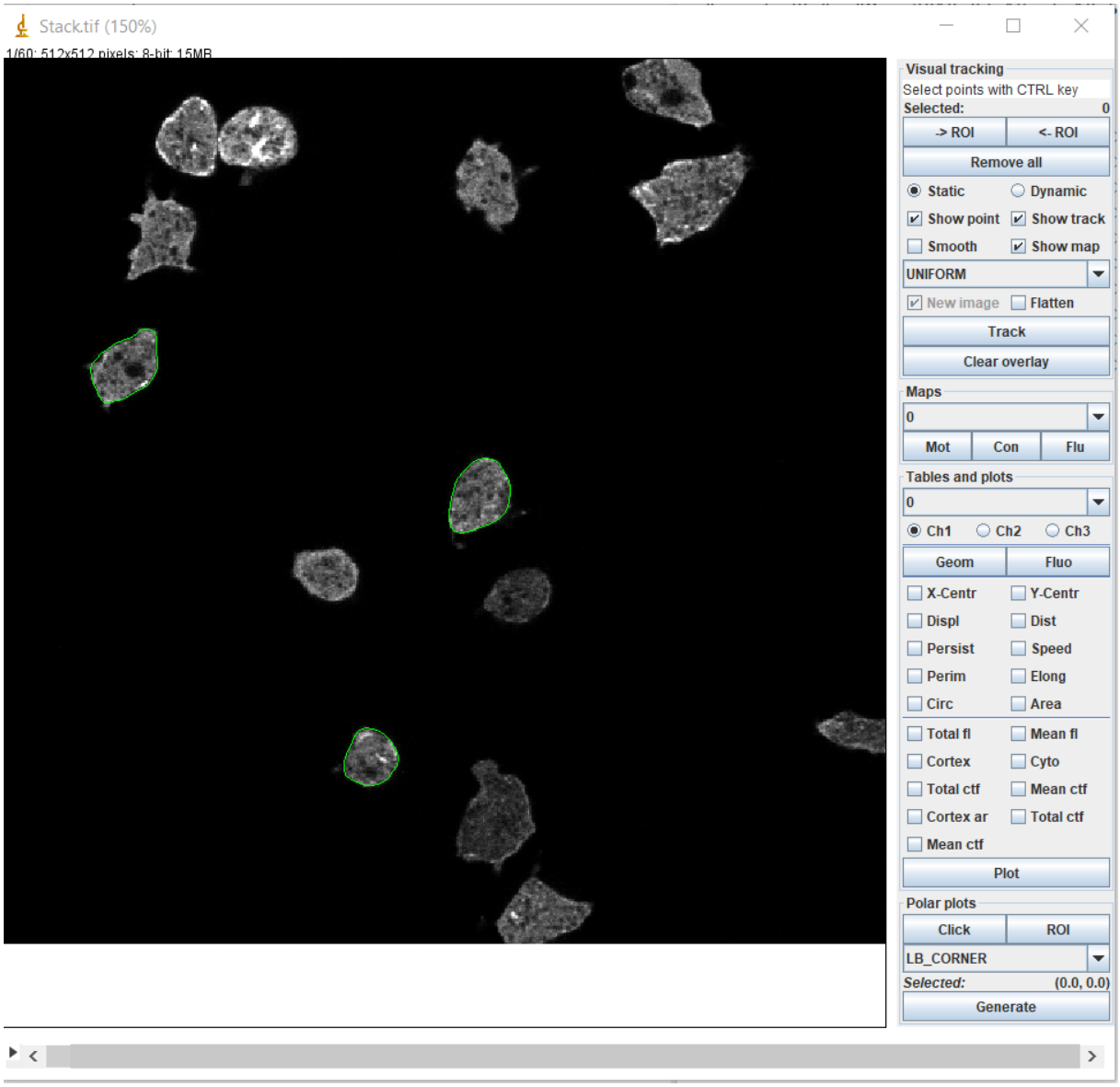


Figure 14: Protrusion Analysis module dialog

There are four sections on the left side of the user interface:

1. **Visual tracking** - allow to define points of interests either directly on the cell contour or

import them from ROI manager. See note below about coordinate systems and importing from ROI scenarios. Tracking also produces *tracks.csv* files with tracks.

2. **Maps** - display *Motility*, *Convexity* or *Fluorescence* maps for selected cell.

3. **Tables and plots** - produce IJ Result tables with statistics for selected cell (*Geom* button) or fluorescent channel (*Fluo* button). These are the same data as included in the *.csv* file generated and updated by QuimP pipeline (BOA, ECMM, ANA, Q-Analysis).

4. **Polar plots** - generate and save in current folder polar plots. Polar plot represents local velocity of the cell membrane averaged over time for each node (owing ECMM algorithm, the cell contour can be interpolated to any real number of nodes that can be tracked forward and backward in time). Point on the contour that is closest to location of chemotactic gradient source is considered as the first one at polar coordinate $\theta = 0°$. Chemotactic gradient source can be selected by mouse (*Click* button), from ROI (*ROI* button - first point of the ROI is considered as source) or from predefined settings (screen corners).

13.1 Coordinates and conversions

The contours of the cell consist of vertices, internally each vertex v for cell c is represented by two coordinates: position along outline and time (frame), $v =< p; t >$. The same coordinate system is used for maps generated by Q-Analysis (**Maps** section of the UI). The vertical axis represents time, with frame 1 at the top. The horizontal axis represents positions along the cell outline, the length of which has been normalised to 1 at every frame (see subsection 14.5). Resolution of horizontal axis can be configured in Q-Analysis module (default value is 400). Resolution of vertical axis equals number of frames in the sequence. These both values are stored in the *QCONF* file.

If points selected on the cell contour in the main view of the plugin are exported to ROI manager, they are converted to screen coordinates using configured resolution for positions along the cell outline and number of frames in the sequence (e.g. for default resolution of 400 and 100 frames, point selected at frame 50 in the middle of the contour will have coordinates $x = 200$, $y = 50$). Additionally cell number is encoded in the ROI name as suffix *cell_N_X*, where N is the cell number and X stands for point number.

Reverse operation - importing points from ROI expects the same naming convention unless the user will press and hold *CTRL* key to import all points to the outline of the cell 0. ROI coordinates also should follow guidances given above, they will be converted to ¡position;frame¿ on the outline. Position is selected as the closest vertex on the outline to the input point. Not that ROI can be also shapes, which allows to select more than one point at once.

13.2 Generate tracks

Tracks are generated from selected points on the cell outline. Selections can be made directly on the outline or imported as ROIs.

13.2.1 Manual selection

1. Hold *CTRL* and click points of interest on the outlines. You can select points on different cells and different frames. Use *Remove all* button to delete all selected points.
2. Click Track to see forward (green) and backward (yellow) tracks. Additional options affect how the output information is presented to the user. For details see tooltips.
3. Investigate *tracks_N.csv* in the root folder. It contains indexes that index maps (subsection 14.5) and can be used for extracting e.g. values of motility or fluorescence, e.g. one can use Format Converter to generate *csv* representation of maps (see subsubsection 14.1.1 and subsection 6.2).

13.2.2 Importing from ROI

In this scenario we are interested in detecting protrusions and tracking them. We use motility map to find out points with highest velocity and then we will track them in the plugin.

1. Load *QCONF* file ti the plugin.
2. In **Maps** section select cell and click *Mot* button to open motility map.
3. From Fiji open `Process`→`Find Maxima` plugin and try to identify points in the motility map of highest local velocity (you might need to select options: *Preview*, *Light background* and use threshold around 50). Click Ok if you are happy with results.
4. Add points to the ROI manager (Ctrl+T).
5. Return to the Protrusion Analysis module and click *¡-ROI* to import points to program. Note that by default points are assigned to cell 0, if it is another cell take care about proer naming of the ROI in ROI manager.
6. Click Track to see forward (green) and backward (yellow) tracks. Additional options affect how the output information is presented to the user. For details see tooltips.
7. Investigate *tracks_N.csv* in the root folder. It contains indexes that index maps (subsection 14.5) and can be used for extracting e.g. values of motility or fluorescence, e.g. one can use Format Converter to generate *csv* representation of maps (see subsubsection 14.1.1 and subsection 6.2).

14 Data Files Explained

14.1 Parameter files - new format vs. old format

The latest QuimP introduces a new file format called QCONF[13]. The *QCONF* is based on human-readable JSON format and it is designed to hold all results evaluated by QuimP on every stage of data processing, together with configuration parameters, algorithm settings, etc. (check subsubsection 14.1.2 for details on QCONF structure). Initially, the *QCONF* is generated by BOA plugin together with *stQP* file (the same statistical measurements are also contained in *QCONF*, *stQP* file is saved separately for convenience sake). Remaining modules depend on *QCONF*, thus they read and write all data from/to this file. The *QCONF* is not replacement for "old" formats - it rather extends them and allows for easy sharing of results between different QuimP modules, restoring work from last point and keeping the whole QuimP configuration in one place.

Currently, QuimP understands both formats, but its behavior depends on the status of tick box available in QuimP Toolbar (Figure 2) and on format loaded to module:

- *Use new file format* is selected - only *QCONF* files are shown in file chooser[14]. A *QCONF* loaded to QuimP module will be updated and saved back. Files in previous format (*paQP*, *snQP*, *maQP*, *stQP.csv*) **are not** saved but they can be optionally regenerated by Format Converter[15] in the same directory. **This is recommended approach.**
- *Use new file format* is deselected - both configuration files (*paQP* and *QCONF*) are shown in file chooser. The *paQP* files are generated every time, even if user loaded *QCONF* file. The *QCONF* file **is not** updated or generated if user loaded *paQP*.

Relations between module and file are summarized in Table 1. Legend for Table 1 is as follows:

- **N** - file is created.
- **M** - file is modified - new data are appended to it.
- **C** - file is changed - some data are replaced.

Note
The *QCONF* file can be loaded to Matlab using one of the free JSON parsers (tested with JSONlab[16]).

14.1.1 Conversion between formats

QuimP supports conversion between old *paQP* and new *QCONF* formats. Simple tool for this purpose is available from the QuimP bar (see Figure 2) under *Tools* menu or in extended version as separate module (see Figure 15). The module additionally allows to extract numeric data from *QCONF* and save them in user friendly *csv* format.

[13] above QuimP11
[14] This feature may not work correctly on Windows systems
[15] Available from QuimP Toolbar *Tools* → *Format Converter*
[16] http://uk.mathworks.com/matlabcentral/fileexchange/33381-jsonlab--a-toolbox-to-encode-decode-json-file

Table 1: Relation among modules and generated files

	paQP	snQP	maQP	stQP.csv	QCONF	pgQP	svg	tiff
BOA	N	N		N	N	N		
ECMM		C			M			
ANA	M	M		M	M			
Q Analysis			N		M		N	N

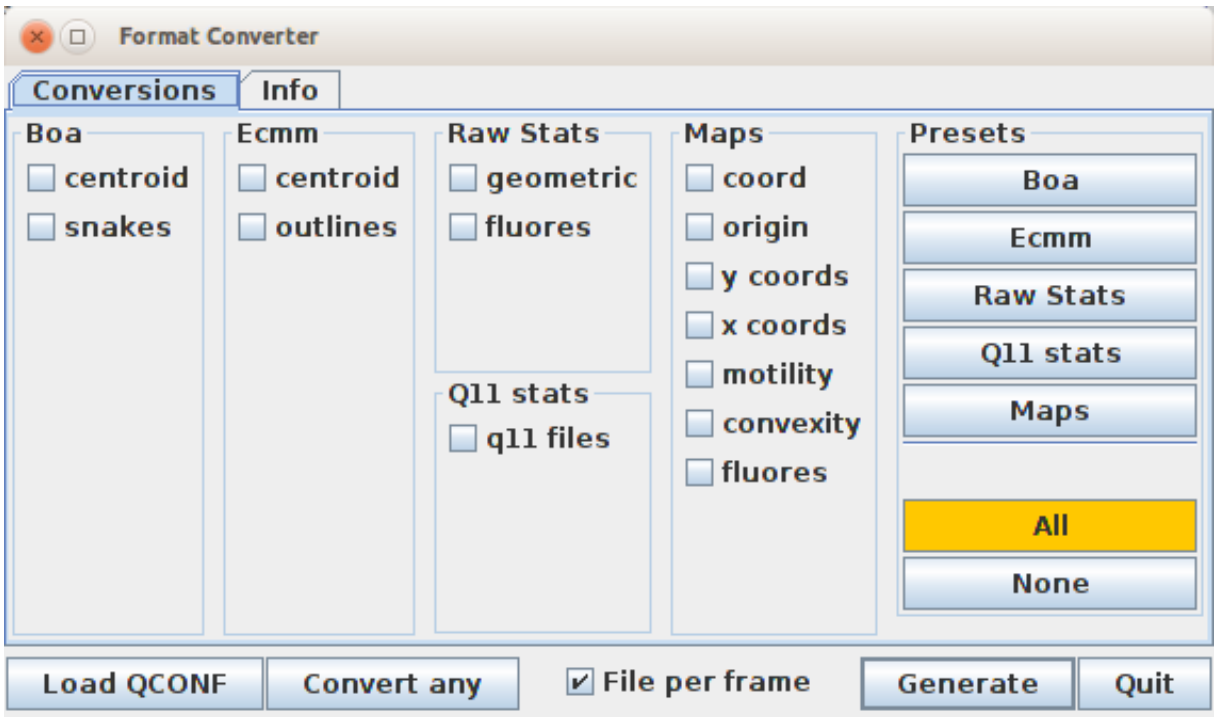


Figure 15: Format converter dialog.

Conversion between formats - *Convert any* button

Conversion tool is available from either the QuimP bar or Format Converter module under *Convert any* button. After selecting proper input file - *QCONF* or *paQP*, the opposite format is saved in the same folder. The following rules will apply:

1. If *paQP* is an input file, it must be the first one with suffix *_0.paQP*.
2. The *QCONF* format stores separate segmentation settings for each frame - only settings from the first frame are saved in *paQP*.
3. Order of outline nodes in *QCONF* can be randomly shifted forward or backward by one node during conversion from *paQP*.
4. *paQP* format stores data with precision limited to two decimal places.

Extracting data from *QCONF* format - *Load QCONF* button

Format converter module (Figure 15) allows to save selected parameters evaluated during QuimP workflow and stored in *QCONF* file as separate *csv* files. Some parameters (like e.g. those under

Raw Stats) coincide with content of *snQP* or *stQP.csv* files. Data extraction is applicable only to *QCONF* files that should be loaded by *Load QCONF* button. Conversion process is initiated by *Generate* button. Output files are saved in the same folder as input *QCONF*.

14.1.2 Processing data files in other languages

Simple example of processing QuimP data files in Python is available at our GitHub account[17].Notebook contains also scheme of *QCONF* structure generated by tool available in the same repository.

Similar approach can be used in any language supporting reading JSON files. One can also convert *QCONF* to old file format, based mostly on widely understood *csv* files, and follow e.g. this example in Matlab[18]

14.2 Parameter Files - *paQP* Files

A parameter file contains text and has the extension *.paQP*. Both QuimP's plugins and Matlab functions use the parameter file to locate files associated with an analysis and also to store data such as the pixel size, frame interval, and start frame. In addition, it stores the parameters used for segmentation (the last used parameters).

The first line identifies the file as a QuimP parameter file, and contains the creation date. The second is random identifier. The third is the absolute path to the image used for segmentation.

In general, the contents of a parameter file should not be changed, but you may wish to alter the paths to image files if you later move them.

14.3 Snake Data - snQP files

The *.snQP* file contains data relating to the nodes of cell outlines (*snake* is a reference to active contours being described as snakes). Comment lines begin with a #. Data for frame f begins with the comment line *#frame f*, followed by a single integer value indicating the number of nodes that make up the outline (N). The following N rows hold the data for the nodes. The columns are as follows:

- **Node Position** - The normalised position of the node in relation to node zero, n_0 (see subsection 9.1).
- **X_coord** - The horizontal pixel coordinate of a node on the image used for segmentation.
- **Y_coord** - The vertical pixel coordinate of a node on the image used for segmentation.
- **Origin** - The position from which a node originated from on the outline at the previous frame (see subsection 9.1).

[17]https://github.com/CellDynamics/QuimP-Python/blob/master/qconf_examples.ipynb
[18]https://pilip.lnx.warwick.ac.uk/docs/Walkthrough.html

- **Global Origin** - The position from which a node originated from on the outline at the first segmented frame (see subsection 9.1).
- **Speed** [microns per second]- The speed at which a node travelled between frames, as determined by ECMM.
- **Fluor_Ch$** - The intensity value assigned to a node by ANA (see section 10) for the channel number denoted by $. This value is the maximal $3x3$ average intensity measured at the node's local cortical region.
- **Ch$_x** - The horizontal pixel co-ordinate on the fluorescence image at the point where Fluor_Ch$ was measured, i.e. the co-ordinate where the maximum intensity occurred.
- **Ch$_y** - The vertical pixel co-ordinate on the fluorescence image at the point where Fluor_Ch$ was measured, i.e. the co-ordinate where the maximum intensity occurred.

  It is possible to store data for 3 fluorescence channels ($\$ = \{1, 2, 3\}$) within the *.snQP* file. Missing data is denoted by negative values.

14.4 Frame statistics - stQP files

The statistics file, with the extension *.stQP*, contains whole cell statistics for each frame (rather than node associated data). Data is written as *comma separated values* which can be opened easily as a spreadsheet. The file is split into two section. Data in the top section is computed by BOA and relates to cell morphology and movement of the cell centroid. Data in the lower section relates to cell fluorescence and was computed by ANA (data is listed separately for all 3 available channels). Missing data is denoted by -1.

The cell centroid is computed as the weighted centre of the polygon formed by the cell outline. The measures computed by BOA are as follows:

This file is generated and updated regardless of selected file format (*QCONF* or *paQP*). Refer to Table 1

- **X-Centroid** [*pixels*] - Horizontal pixel co-ordinate of the cell centroid.
- **Y-Centroid** [*pixels*] - Vertical pixel co-ordinate of the cell centroid.
- **Displacement** [*microns*] - Distance the cell centroid has moved from its position in the first recorded frame.
- **Distance Travelled** [*microns*] - Sum of the centroid displacements between frame, i.e. the total distance over which the centroid has moved.
- **Directionality** - Persistence in direction, calculated as $Displacement/Dist.Travelled$ (chemotaxis index). A value of 1 reveals that a cell has moved in a straight line. Decreasing values denote a cell moving increasingly erratically.
- **Speed** [*microns per second*] - Speed at which the centroid moved between the current and previous frame.

- **Perimeter** [*microns*] - Length of the cell perimeter (segmented outline).
- **Elongation** - An ellipse is fitted to the cell outline and the major/minor axis used to compute the elongation of the cell's shape. $Elongation = major\ axis/minor\ axis$. Note that a value of 1 does not necessarily represent a perfectly circular cell, only a circular fitted ellipse.
- **Circularity** - A measure of circularity defined by the following equation: $\frac{4*PI*Area}{Perimeter^2}$. A value of 1 reveals the cell's outline to be perfectly circular.
- **Area** [$microns^2$] - Cell area.

The measures computed by ANA are as follows:

- **Total fluo.** [*pixelintensity*] - Total fluorescence. Sum of all pixel intensities within the cell outline.
- **Mean fluo.** [*pixelsintensity*] - Mean fluorescence. Average intensity of pixels within the cell outline.
- **Cortex width** [*microns*] - Width of the cortex, as specified by the user (see section 10).
- **Cyto. area** [$microns^2$] - Area of the cytoplasm (area of the whole cell minus the cortex area).
- **Total cyto. fluo.** [*pixels intensity*] - Sum of all pixel intensities within the cytoplasm.
- **Mean cyto. fluo.** [*pixels intensity*] - Average pixel intensity within the cytoplasm.
- **Cortex area** [$microns^2$] - Area of the cortex.
- **Total cortex fluo.** [*pixels intensity*] - Sum of all pixel intensities within the cortex.
- **Mean cortex fluo.** [*pixels intensity*] - Average pixel intensity within the cortex.

14.5 QuimP Maps - maQP files

The images produced by Q Analysis are spatial-temporal maps of motility, convexity, and fluorescence. The vertical axis represents time, with frame 1 at the top. The horizontal axis represents positions along the cell outline, the length of which has been normalised to 1 at every frame. Outlines can be thought of as having been 'cut' at position zero, laid along the horizontal axis, and stacked below one another, hence maps are cylindrical (wrap around onto themselves on the horizontal axis).

The node randomly designated as begin node zero (n^0), at frame 1, is mapped to the top left pixel. Data from ECMM is used to track the position of n^0 throughout all the frames, and is always positioned at the left most pixel at each frame. All other nodes are positioned along the horizontal axis according to their distance from n^0 along the cell perimeter. Values between nodes are estimated using linear interpolation.

If the number of frames is below the specified map resolution, then maps are scaled vertically, hence a row of pixels may be an interpolated value between frames.

As well as producing *.tif* images, maps are also saved as text images with the extension *.maPQ*. Unlike the *.tif* maps, these are not scaled vertically, so each line of values contains the data for a single frame. It is these maps that should be used for further analysis.

Similarly, four additional maps are produced to aid further analysis:

- **Motility Map.** Pixels are coloured according to node speed, as calculated by ECMM. Red shades represent expanding regions, blue shades contracting regions. Pixel values within the *tiff* image are scaled to fill the colour spectrum. The map file extended *_motilityMap.maPQ* contains un-scaled values, in *microns per second*.
- **Fluorescence Maps**. A fluorescence map is produced for each channel that data has been recorded into. The fluorescence map images display pixel intensities, as extracted by the ANA plugin. Files extended *_fluoCh$.maPQ* contain the respective data. Fluorescence value is a mean of pixel intensities sampled within 9-point stencil for each node of the cortex outline (see section 10).
- **Convexity Map**. Represents the curvature of the cell, measured in the range $[-1, 1)$, where negative values are concave (blue), and positive convex (red). As with the motility map, values are scaled to fill the colour map. Files extended *_convexityMap.maPQ* contain the respective data (un-scaled).
- **Co-ordinate Map (a.k.a. Position Map)**. As described in subsection 9.1, each node has an associated position. The co-ordinate map, rather than contain values regarding motility, fluorescence or convexity, instead contains the position values of nodes. The main purpose of the co-ordinate map, along with the origin map, is for tracking positions through time (see subsection 16.3).
- **Origin Map**. As described in subsection 9.1, each node has an origin, the position a node originated from on the previous frame. The origin map contains origin values and can be used, along with the co-ordinate map, to track positions through time (see subsection 16.3).
- **xMap**. Contains horizontal image pixel co-ordinates (those on the image used for segmentation) relating to map pixels.
- **yMap**. Contains vertical image pixel co-ordinates (those on the image used for segmentation) relating to map pixels.

14.6 Scalable vector graphics output - svg files

Files extended with *.svg* can be viewed natively in most internet browsers (Windows Explorer requires the Adobe SVG viewer), or opened and edited in graphics software (e.g. Inkscape, free software for mac). The Scalable Vector Graphics (svg) file format can be viewed at any desired resolution.

- **Cell track**. An overlay of all cell outlines coloured according to frame number. Files extension *_track.svg*.
- **Cell motility movie**. Data contained within the motility/fluorescence/convexity map is displayed on the cell track. Extension *_motility.svg*.

14.7 BOA plugins stack configuration - pgQP file

This file stores configuration of plugin stack together with configurations of active plugins (see Figure 4 and subsection 8.3). This file is not related to any loaded data therefore it can be exchanged between projects. Complex plugin stack configuration is stored also in *QCONF* file.

15 ImageJ macro support

Most of QuimP modules support ImageJ macro language but syntax of the command (especially parameters passed to the module) differs from that commonly used by other ImageJ plugins. Use Macro Recorder tool to discover parameters supported by plugin. Note that any file name or path should be enclosed in parentheses if it contains spaces. Windows users should use slash symbol in path names (e.g. c:/Users/myfile.QCONF). Quote character is not allowed in parameter string. Skipped parameters get their default values, therefore output from Macro Recorder can be greatly reduced by removing options that are not used or unchanged.

Note that macro strings are outputed to Macro Recorder only if plugin is run from Menu. **It does not work if plugin has been started from QuimP Toolbar.**

16 MATLAB Functions

Archive with Matlab routines mentioned in this section is available on QuimP web page (section Download QuimP)[19]

Matlab routines work with old data format only (**.paQP*, **.snQP*, etc.). To use them with *QCONF* file, it has to be first converted to the old format by Format Converter (section 14). It is also possible to load *QCONF* to Matlab using JSon readers.

QuimP outputs a large amount of data for each cell analysed. You may want to write custom scripts for analysis, but we provide a set of MATLAB functions to remove the hassle of loading/organising data, and performing simple operations, such as plotting maps. We hope to make data as accessible and as explorable as possible to allow adaptation to a users specific goals.

Help documentation is included with each function which can be accessed in the usual way by typing *help functionName* at the MATLAB command prompt. At the end of this document is a walkthrough analysis using the provided test images (see subsection 6.1).

16.1 Loading data - *readQanalysis.m*

The first step is to launch the script *readQanalysis.m* which handles loading of data from multiple cells. Given a directory, it will search that directory, and all sub-directories, for *.paQP* files. For each parameter file that is found an analysis structure is built. As data is read from sub-directories, you may organise your separate analyses into sub-directories, but placing all files into the same directory is still possible. Missing files are skipped, hence it is not necessary to have run all the QuimP plugins.

Loading data relies on the maintenance of file names imposed by the QuimP plugins, and requires associated files to be present in the same directory as the parameter file. The exceptions to this are the images used for segmentation and fluorescence measurements. Paths to these images are stored in full to allow separation of analysis files from image data. If images are moved, and subsequently cannot be located, the directory where the parameter file is stored will be searched. You may edit the *.paQP* file to alter paths to the images if you wish (note that images are not loaded into memory by *readQanalysis.m*, instead the function checks only that they exist and sets a path to them).

The output of *readQanalysis.m* is an array of structures, C, each element holding data for a single analysed cell. The contents of an analysis structure is as follows (F = number of frames, N^f = number of nodes at frame f):

- **name** [*string*] - File name element common to all associated files (name of the parameter file).
- **index** [*integer*] - Cell ID, in order read in.
- **PATH** [*string*] - Directory of the parameter file.
- **$FILE** [*string*] - Filenames of QuimP output.

[19] http://www2.warwick.ac.uk/fac/sci/dcs/people/till_bretschneider/quimp/test_data

- **$TIFF** [*string*] - Filenames of an image sequences.
- **outlines** [*cell array* $(F \times 1)$] - Holds node associated data. Each element contains a matrix of size $N^f \times 6$, each row being a node. Column headers are [Position , X-coord , Y-coord , Origin , Global origin , Speed].
- **outlineHeaders** [*cell array* (1×6)] - Descriptions of column measures within *outlines*, as stated above.
- **fluo** [*cell array* $(F \times 1)$] - Holds node fluorescence data for all three channels. Each element contains a 3D matrix of size $N^f \times 3 \times 3$. The first dimension represents nodes, second the channel number, and third the measure. The measures are [Intensity , X-coord , Y-coord], where the x and y co-ordinates provide the location of sampling (see section 10). For example, the intensity sampled at node 23, on channel 2 is located at (23,2,1).
- **fluoHeaders** [*cell array* (1×3)] - Descriptions of measures within *fluo*, as stated above.
- **nbFrames** [*integer*] - The number of frames.
- **startFrame** [*integer*] - The first frame segmented relative to the the image used for segmentation.
- **endFrame** [*integer*] - The last frame segmented relative to the the image used for segmentation.
- **frames** [*vector* $(F \times 1)$] - Vector of the segmented frames relative to the the image used for segmentation.
- **PixelSize** [*double*] - Pixel size in microns (image scale).
- **frameInterval** [*double*] - Frame interval in seconds.
- **R** [*vector* (1×4)] - Pixel co-ordinates of a bounding box encompassing all movement of the cell. R = [x min, x max, y min, y max].
- **maxSpeed** [*vector* $(F \times 1)$] - For each frame the maximum node speed is calculated. This is useful for scaling plots.
- **stats** [*matrix* $(F \times 11)$] - Holds whole cell statistics relating to the cell centroid and cell shape (see subsection 14.4). Rows contain data for frames. The column headers are ['Frame' 'x-Centroid' 'Y-Centroid' 'Displacement' 'Distance Travelled' 'Directionality' 'Speed' 'Perimeter' 'Elongation' 'Circularity' 'Area']
- **statsHeaders** [*cell array* (1×11)] - Descriptions of column measure within *stats*, as stated above.
- **fluoStats** [*matrix* $(F \times 3 \times 11)$] - Holds global cell statistics relating to the cell fluorescence (as described in subsection 14.4). The first dimension is frame number, second the channel number, and third the measure. The measure headers are ['Frame' 'Total Fluo.' 'Mean Fluo.' 'Cortex Width' 'Cyto. Area' 'Total Cyto. Fluo.' 'Mean Cyto. Fluo.' 'Cortex Area' 'Total Cortex Fluo.' 'Mean Cortex Fluo.' '%age Cortex Fluo.']. Missing data is represented by negative values, typically -1.

- **fluoStatHeaders** [*cell array* (1×3)] - Descriptions of the measures in the third dimensional within *fluo*, as stated above.
- **cortexWidth** [*vector* (3×1)] - Holds for each channel the cortex width specified when running ANA.
- $*$**Map** [*matrix* $(F \times MapRes)$] - Maps read from *.maQP* files (see subsection 14.5). A QuimP map in MATLAB is a 2D matrix, rows being frames, columns being discrete points along the cell outline.
- **forwardMap** - see subsection 16.3.
- **backwardMap** - see subsection 16.3.

16.2 Function Overview

Functions prefixed with 'read' are used by *readQanalysis.m* and so generally are not used independently, but can be if desired. For details of usage please refer to the functions help by typing *help functionName* at the MATLAB command prompt.

- **buildTrackMaps** - Constructs the *forwardMap* and *backwardMap* based upon the co-ordinate and origin maps. These are used to track through QuimP maps. See subsection 16.3 for details.
- **mapLookup** - Extract values from maps by specifying a window or region. This can be used, for example, to extract data along tracked paths.
- **plotMap** - Plot a QuimP map using specified colour map limits.
- **plotMotility** - Plots a cell outline with nodes coloured according to their speed of movement.
- **plotOutline** - Plots a cell outline.
- **trackBackwards** - Utilises the *backwardMap* to track backwards through a map, from a specified point, in accordance with the ECMM tracking data.
- **trackForwards** - Utilises the *forwardMap* to track forwards through a map, from a specified point, in accordance with the ECMM tracking data.
- **trackForwAcc** - Similar to *trackForwards* but functions at sub-pixel resolution. More accurate but slower.
- **xcorrQ** - Perform cross-correlation and auto-correlation of .maPQ maps.

16.3 Tracking Maps

The term *tracking* in this sense refers to being able to pick a position on the perimeter of a cell and identify its corresponding position in the following frames, and hence, track a position over time. The trace of a position, computed by ECMM, can be plotted on top of a QuimP map.

A QuimP map in MATLAB is a 2D matrix, rows being frames (f) and columns being discrete locations along the cell outline (l). An incorrect view may be that a map pixel p at frame f, and location l, (p_l^f), tracks to the pixel directly below (p_l^{f+1}). This is not the case, p_l^{f+1} does not necessarily originate from p_l^f. The origin map must be consulted to identify the correct location.

[Maps that enforce p_l^f tracking to p_l^{f+1} become heavily distorted, and certain regions lose resolution severely.]

To track from p_l^f to p_{l2}^{f+1} we require the correct matrix column index at $f+1$, $l2$. Firstly, consult the co-ordinate map at p_l^f to identify the position on the cell outline. Next, extract the values in row $f+1$ from the origin map. The index $l2$ is simply the index of the closest origin value to p_l^f's position (this can also be done at sub-pixel resolution for more accuracy). The process is repeated to find p_{l3}^{f+2}, p_{l4}^{f+3}, etc. In a similar way, positions can be tracked backwards in time.

This tracking procedure is implemented in the provided MATLAB functions. The function *buildTrackMaps.m* creates two additional maps, *forwardMap* and *backwardMap*, and is called when data is read in. An element in the *forwardMap*, for example at p_l^f, contains the correct column index for $l2$ to locate p_{l2}^{f+1}. Similarly, *backwardMap* contains the correct values for finding p^{f-1}.

Functions *trackForwards.m*, *trackForwAcc.m* and *trackBackwards.m*, given a frame, location, and number of frames to track over, will return complete traces for you. Consult the help for these functions for details of usage and the walkthrough for an example.

17 Historical versions

17.1 QuimP11b

1. Fixed several minor bugs.
2. Updated and tested with ImageJ 1.49a and MATLAB 2014a
3. The ECMM and Q Analysis plugins will search for other .paQP in a directory and prompt to batch process files.
4. BOA prompts to check image scale.
5. BOA can read in a previous segmentation using the LOAD button
6. When editing segmentations the user can scroll between frames without leaving edit mode

18 Contact

The QuimP software is being developed at the Department of Computer Science at the University Of Warwick by the Till Bretschneider group (Till.Bretschneider@warwick.ac.uk). Further information, and future releases, can be obtained from `http://go.warwick.ac.uk/quimp`.

QuimP is under continuous development and feedback will be much appreciated. If you discover a bug, have suggestions for features, or simply find a spelling mistake, please contact the developer Piotr Baniukiewicz at p.baniukiewicz@warwick.ac.uk. Thank you for supporting our software.

References

[1] Leo Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, 2006.

[2] Zvi Kam. Microscopic differential interference contrast image processing by line integration (LID) and deconvolution. *Bioimaging*, 6(4):166–176, 1998.

[3] R A Tyson, D B A Epstein, K I Anderson, and T Bretschneider. High Resolution Tracking of Cell Membrane Dynamics in Moving Cells: an Electrifying Approach. *Mathematical Modelling of Natural Phenomena*, 5(1):34–55, 2010.