Specification document (CellShape v0.8)

[CellShape: an image analysis tool for bacterial fluorescent microscopy – Angel Goñi-Moreno]

1 – Software organisation

CellShape groups a total of nine Python scripts, that are related as depicted in Figure 1. The file `CellShape.py` contains the Graphical User Interface (GUI) description and is the file that launches the application (see Usage). The software has been developed in such a way that an expert user could get rid of the GUI with little modification, as `CellShape.py` imports only one file. However, we strongly recommend to run the tool as presented, specially if the user is not an advanced programmer.

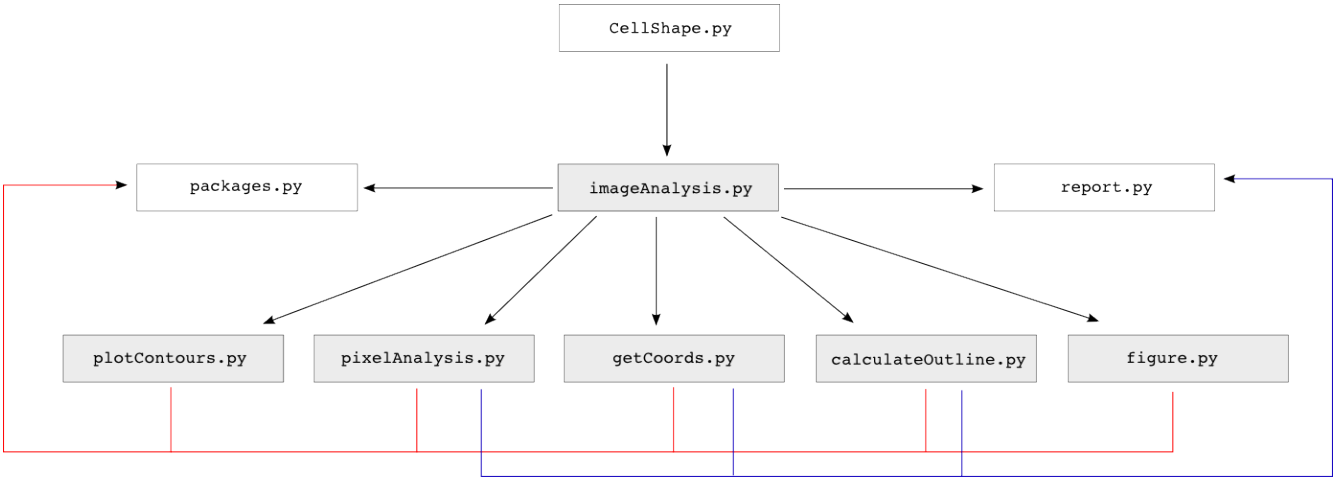


Figure 1. Organisation flowchart. The arrows mean that the source file imports the target file. Shadowed files are the core of CellShape, where all major functions are coded. The file at the top encodes the GUI.

The file `imageAnalysis.py` is called right after the user enters his/her options in the GUI. From now until the analysis process is finished, the application runs like a *black box* hidden to the user. Each file that is imported from this central one, codifies different functions of the image analysis work-flow, specially the files shadowed in Figure 1. The calculation of contour lines from a matrix of floats is done in `plotContours.py`; the conversion of a .TIF image into such a matrix in `pixelAnalysis.py`, the extraction of two-dimensional coordinates of the fluorescent areas analysed for plotting purposes in `getCoords.py`, the selection of those contour lines that best match the shapes are coded in `calculateOutline.py`, and the creation of a new figure in `figure.py`. The Python packages used by CellShape are those found in the `packages.py` file. The informative messages printed out in the GUI are produced by the functions coded in `report.py`.

2 – Functions included in v0.8

The functions included in this version of CellShape are: 1) localisation of mRNA spots and their positioning relative to nucleoid areas, 2) quantification of fluorescent signal in cells (cytometry simulation), 3) correlation of different coloured signals in the same cell, and 4) analysis of a single image. These four functions can be selected by the user using the right hand side column of the GUI (Figure 2).

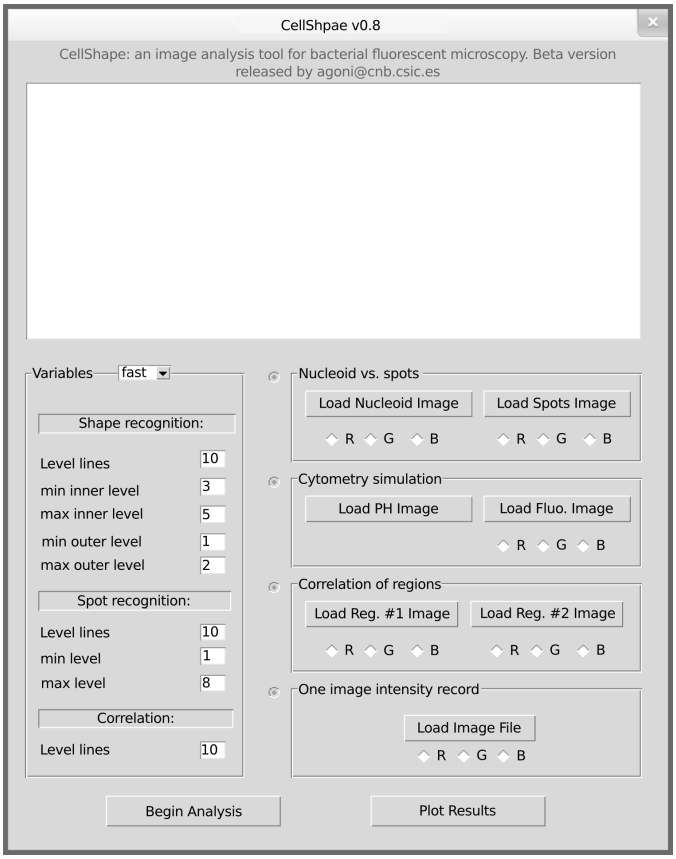


Figure 2. Graphical User Interface. The blank space at the top is the “Information Screen”, where all events of the software are stated. The left column, termed “Variables”, specify the parameters used by the application (explained in section Usage). The right column enumerates the different functions the user can run.

The application lets the user select between the four functions. In each of them, the user can fill the parameters needed for the analysis to begin: the images must be loaded and the colour channel selected (R: red, G: green, B: blue). After the function has been chosen, the user should click at the “Begin Analysis” button. Although the functions have been shaped to our needs, the analysis can be re-directed to other purposes by selecting the appropriate variables (see section Usage).

3 – Usage

To start using CellShape, the user has two options: 1) run the executable file (double-click to run), or 2) launch the code from a terminal by writing “$ python CellShape.py” inside the folder with the Python packages. While the first option is quicker, the second one starts a faster application. However, launching the software from the terminal requires to have a Python installation (by default in UNIX-based systems) with the modules specified in the `packages.py` file. Namely: `PIL` (pillow), `matplotlib`, `numpy`, `shapely` and `tkinter` (see Appendix). CellShape is entirely Python powered so there is no need to install any other third-party code. Depending on system specifications, the GUI could look slightly

different in several ways. The GUI of Figure 2 is the product of running the software on a Debian-7 machine.

As it is impossible to define universal values for all pictures (due to different light sensitivity, quality or definition), we suggested the parameters under the “Variables” section to be accurate enough to customize the analysis. CellShape is based on the calculation of level lines on a surface of pixel values. That is to say, an image is transformed into a matrix of pixel values (depending on the RGB channel selected) and then segmented using contour lines to identify shapes. The number of contour lines to use and the selection of the most relevant ones, have to be adjusted depending on user needs; this is done by changing the Variables section.

What is the meaning of the variables under the Variables section? Figure 3 shows the link between the variables and the contour lines, which is the key feature to fine-tune the software.

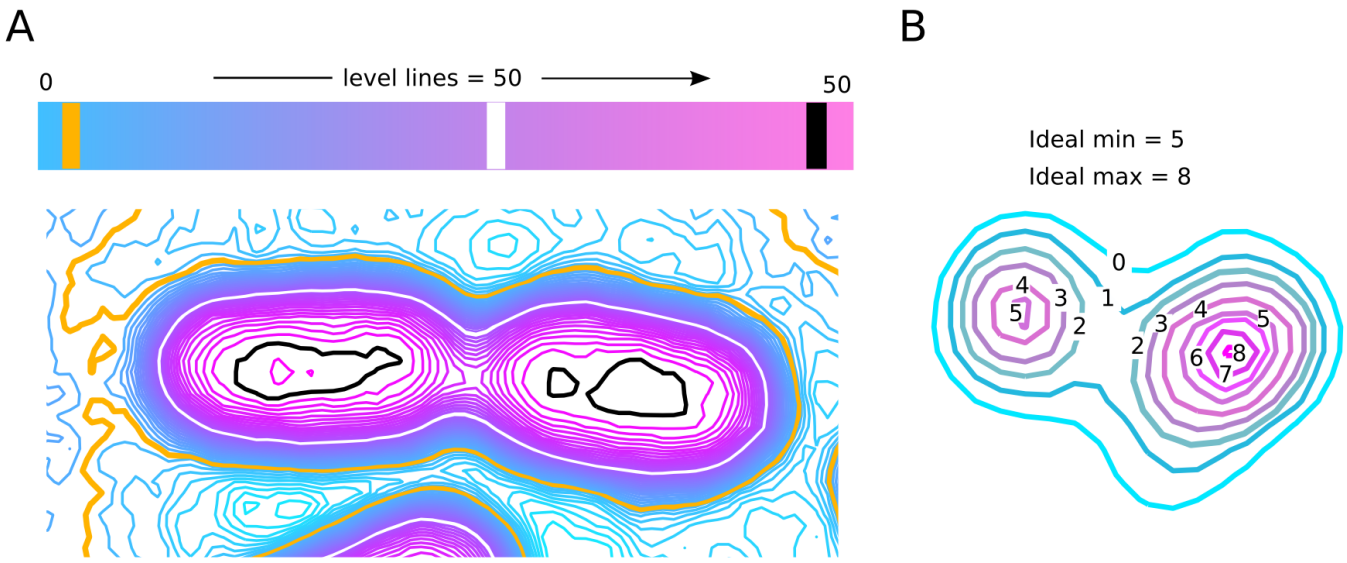


Figure 3. Meaning of the variables. A) Level map as a result of segmenting a matrix of pixels with 50 contour lines. Three levels are coloured to highlight their position. B) Level lines in spot recognition.

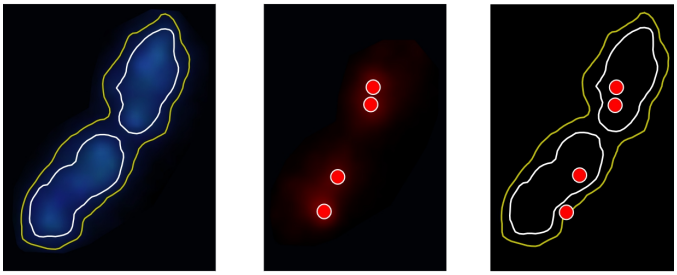
The number of `Level_lines` tells the software how many lines to use in the segmentation process; the larger this number is, the more accurate the selection process results. In Figure 3A we show an image (phase contrast, in this case) segmented with 50 level lines. If we want CellShape to select the contour of the cell, we need to specify the correct lines. If, for instance, the number of the line to select is very low (yellow in Figure 3A), the outline of the cell will not be detected correctly (“open” cell). A very high value will not work either (black line in Figure 3A) as the software may return small inner regions. However, a good selection (white line in Figure 3A) would result in the perfect outline.

Instead of having to specify a single threshold-like value, CellShape allows for the definition of intervals between a `min_level` and a `max_level`. The advantage of this approach is to use several thresholds within a single image to record several intensities. Figure 3B shows the meaning of the intervals using two fluorescent spots, which are levels 5 and 8. On the one hand, if we use a threshold of 8 we will detect only one spot. On the other hand, if the threshold is 5, we will not be accurate enough in one of the spots. The solution is to define an interval [`min_level`, `max_level`] and let CellShape detect the spots in the `max_limit` first and then in the previous one until reaching the `min_level`. All levels outside the defined interval are not considered in the analysis.

One of the main features included in this CellShape distribution is to recognize nucleoid areas, for which we defined inner and outer regions. As the refraction caused by the microscope varies, we found useful to consider two different nucleoid areas. Furthermore, they help the user to obtain statistics about the localization of the spots. Although variable definition should be done for any individual case, we provided three set of values (fast, deep, shape) to help new users of CellShape.

4 – Examples

A) "Nucleoid vs. Spots"



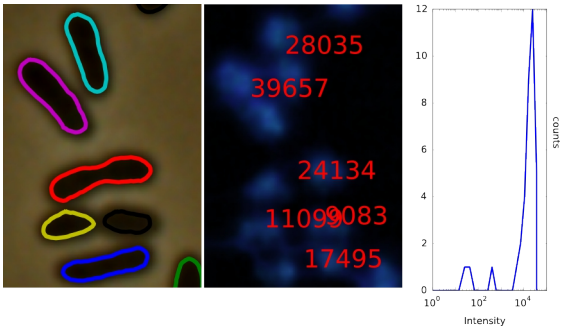
In this example we check where the mRNA spots are placed regarding the nucleoid of the cells (*P.putida*). First of all, select the circle-shaped button next to "Nucleoid vs. Spots" (it is selected by default in the executable Mac-OS and Windows versions). Then press on the button "Load Nucleoid Image". Browse to the folder provided in the CellShape distribution called "Nucleoid-vs-Spots" and select the "Nucleoid.tif" image (file "Nucleoid2.tif" is another example of the same function). Now, back to the CellShape screen, select the "B" button in the RGB (red, green, blue) list below the "Load Nucleoid Image" button. This is because the nucleoid areas are DAPI-stained, so the fluorescent signal is blue. Now load the "Spots.tif" file found in the same folder, and marked in red, "R".

Before running the application, select the set of variables termed "fast" (fast processing) from the drop-down menu next to the "Variables" label. Now click on the button "Begin Analysis" and, once the word "Done!" appears on the information screen, click on "Plot Results". If everything went OK, four windows will pop up with the results:

1 – Contour (level) Lines. Plot with the analysis of the pixel-based image by grouping intensities in levels.
2 – Contour recognition. Output of applying the max-min parameters on the contour lines for shapes. The original image is over-imposed for visualization purposes.
3 – Spot recognition. Output of applying the max-min parameters on the contour lines for spots. The original image is over-imposed for visualization purposes.
4 – Spots in nucleoid regions. Mixture of both shapes and spots.

A few statistics of the image are written on the information screen. Try the same example with the sets of variables "deep" and "phase" to see differences.

B) "Cytometry simulation"



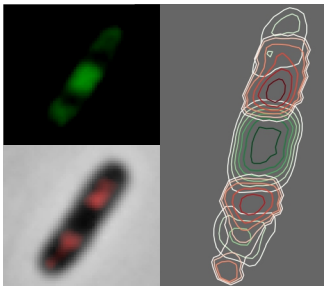
In this example we obtain quantitative information about the fluorescence intensity of cells (*P.putida*). First of all, select the circle-shaped button next to "Cytometry simulation". Then press on the button "Load PH Image". Browse to the "Cytometry-simulation" folder and select the file "Phase.tif". Then load the file "Nucleoid.tif" (placed in the same folder) as the "Fluo. Image" and select the "B" button from the three R-G-B options. This cells are DAPI-stained so their nucleoid display blue fluorescent signal.

Before running the application, select the set of variables termed "phase", explicitly included for this case. Now click on the button "Begin Analysis" and, once the word "Done!" appears on the information screen, click on "Plot Results". If everything went OK, five windows will pop up with the results:

1 – Contour (level) Lines. Plot with the analysis of the pixel-based image by grouping intensities in levels.
2 – Cell recognition. Segmentation of a phase-contrast image to identify individual cells.
3 – Intensity map. The original fluorescence image where the cells are tagged with a number that is the sum of its pixel intensities.
4 – Cytometry Emulator. The record of all intensities (x axis) and the number of cells with that value (y axis).
5 – Cytometry Emulator - log. The same information as before but with a logarithmic x axis and intervals

for counting the cells (y axis) in them.

C) "Correlation of regions"

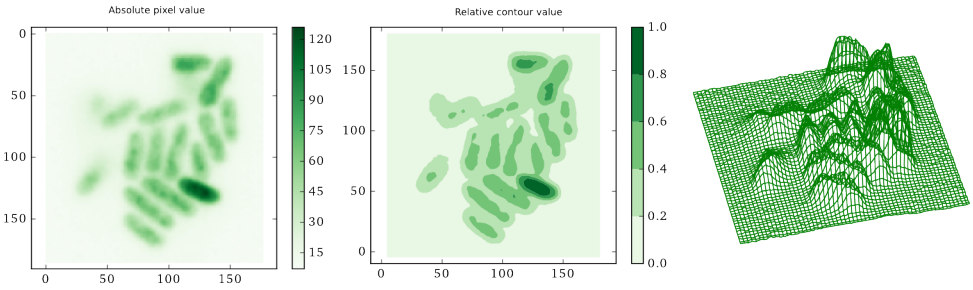


In this example we study the spatial correlation of two regions within the same cell (*P.putida*). First of all, select the circle-shaped button next to "Correlation of regions". Then press on the button "Load Reg. #1 Image". Browse to the "Correlation-Regions" folder and load the image "Reg1.tif". Select the "G" button just bellow, in the R-G-B options, as this region is marked in green. Then load the image "Reg2.tif" as the second region, and select the "R" colour.

Click on the button "Begin Analysis" and, once the word "Done!" appears on the information screen, click on "Plot Results". If everything went OK, five windows will pop up with the results of the analysis:

1 – Region 1 pixel view. Quantification of each pixel intensity. The colour-bar has a relative scale (0..1) where 0 is the lowest intensity found in the image and 1 is the highest.
2 – Region 2 pixel view. The same as above but with the other region.
3 – Region 1 contour field. Contour lines of the first region intensity.
4 – Region 2 contour field. Contour lines of the second region intensity.
5 – Combined contours. Overlap of both regions.

D) "One-image intensity record"



In this example we analyse a single image to obtain quantitative information. Select the circle-shaped button next to "One-image intensity record" and load the image "Single.tif" found in the provided folder called "One-Image". Choose the "G" option, as it is a green-coloured image, and click on "Begin Analysis". Once the word "Done!" appears on the information screen, click on "Plot Results". If everything went OK, four windows will pop up with the results:

1 – Pixel view (relative). Quantification of each pixel intensity. The colour-bar has a relative scale (0..1) where 0 is the lowest intensity found in the image and 1 is the highest.
2 – Contour view. The same information as above but in a contour-line plot style.
3 – Pixel view (absolute). Quantification of each pixel intensity. The colour-bar has an absolute scale (0..255) where each intensity corresponds to the microscope measurement.
4 – 3D Mesh view. Three-dimensional view of the pixel intensity map in its relative form.

5 – Contact

In order to develop further versions of CellShape, we suggest potential users to tell us the functions they would like to see in the application that are not currently included. If CellShape does not work properly or you discover any bug, please also let us know to improve this distribution. Our contact information is found at http://www.cnb.csic.es/~synbio/

6 – Appendix

In the majority of Linux systems Python has a default installation with most of the packages included but `shapely`. The easiest way to install it is by firstly installing its requirement `$apt-get install libgeos-dev` on Debian/Ubuntu and then `$pip install shapely` on a terminal window. For further instructions about other operative systems please refer to https://pypi.python.org/pypi/Shapely

Under Windows we strongly recommend the use of WinPython from http://winpython.sourceforge.net/ If any problem installing Python or its dependencies, please contact us.

The code has been tested with Python versions 2.6, 2.7 and 3.x.