Preface

Introduction

Please consult this guide before using the Automated Cell Counter Algorithm (ACCA)! It is designed to give the user a brief overview of the necessary installation steps, various counter pipelines and customization procedures.

Contact Information

Ivan Kondratyev (Ikondrat@bu.edu): Please use this address to share any issues, ideas or suggestions for the algorithm. It is still very much work in progress, so all input is generally appreciated. Also, use this address for any counter-specific questions.

Kelton Wilmerding (lwilmerd@bu.edu): Please use this address for any lab-related questions and issues, as well as for all other communications.

Important Announcements

ImageJ v1.53h is broken internally and would not allow manual ROI selection. Please roll back to 1.53g for the ACCA to work properly.

Installation

ImageJ

The ACCA was developed for the ImageJ image analysis software program and it is written in IJ macro language. In order to use the algorithm, you first need to install ImageJ. Please consult this link for the information on how to install it: https://imagej.nih.gov/ij/docs/install/index.html .

Installing the Automated Cell Counter Algorithm

Simply download the cell counter macro file (with extension .ijm) into the Fiji.app/Plugins folder.

Necessary Updates and Plugins

There are a number of plugins that need to be installed for the algorithm to work:

1) ImageJ needs to be updated to version **1.53g** or later. To do so, open the ImageJ control panel and use Help > UpdateJ > 1.53g (Or higher). Run the update and reload ImageJ.
2) 3D ImageJ Suit needs to be installed. To do so, open the ImageJ update control panel by clicking Help > Update from the main control window. Within the ImageJ updater window click on Manage Update Sites and check the box on the 3D ImageJ Suit line. Then click apply changes, wait for all installations and updates to finish. Reload ImageJ and everything should be installed.
3) For manual cell count confirmation you may need to install the Cell Counter Plugin. Please consult this link to do so https://imagej.nih.gov/ij/plugins/cell-counter.html .\

Running the Automated Cell Counter algorithm:

a) Option 1: Use File > Open to locate and open the Automated Cell Counter Algorithm. Then click Run in the lower left corner to run the algorithm or use CNTRL+R.

b) Option 2: Use the search bar in the bottom right of the main ImageJ window to search for the Automated Cell Counter Algorithm and select Run.

Input Formatting

The ACCA is currently limited to processing a very specific format of images. Regardless of their extension, the images should be **2D multichannel composites**, for the ACCA to process them correctly, as it has built-in preprocessing routines to split channels.

Functionality

Currently, the ACCA supports 3 distinct processing pipelines, discussed below:

Automatic Counting

This is the main pipeline of the algorithm. It uses a set of methods to pre-process, segment and count the cells of interest. Currently, it is configured to work with Dapi, cfos, GFP (eYFP) and Overlap counting. The methods can be fine-tuned via an integrated database of tuning profiles (see "methodTuning" function) that can be specified in code for each stain and region. The algorithm relies on users to define the Regions of Interest (ROIs) to be processed, but otherwise can run silently (if appropriate options are selected).

Currently, the Automatic counting pipeline performs the following operations: 1) Enter Batch Mode. This prevents the images from being drawn, greatly reducing the processing time.

2) Analyze the input database (folder).

3) Open each image with the user-specified extension (.oif, .tif, .png or .czi)

4) Create an image Output folder inside the Input folder.

5) Pre-process the image. Split channels and convert it into 8-bit format.

6) Ask the user to specify what channels correspond to what stains.

7) Use user-specified channel designations to save the original image stain channels as .tif files. Construct a cfos x GFP overlap and save it in the output folder.

8) Ask the user to specify (draw) the ROIs to be analyzed. The algorithm processes one ROI at a time, so multiple inputs are required

9) Process each ROI in accordance with the user-defined calibration profile. Utilizes 3D Iterative Thresholding for segmentation and Analyze Particles for counting. Saves Original, Enhanced (auto-contrast function), Thresholded and Auto-Counted images for each stain and region as .tif files. Also saves the counter ROI selections for cfos, gfp and overlap to be used for the Confirmation pipeline.

10) Append the counting results to the Counts master-table. **Please note!** The table is created before the automatic counting pipeline starts, so it will appear blank until an image has been fully processed. Then the counts will be added to the table and it will be refreshed to reflect them.

11) If specified by the user, start a Confirmation and/or Calibration pipeline for the image 12) Close all windows, open the next image

Some useful tips:

- You can use the code to set the "triggerDialogs" variable to "0", which would disable all non-essential dialogs, reducing the amount of required user input to minimum. It is

highly advised to review the user-defined variables block and set it accordingly before disabling the dialogs to avoid issues.

- Please make sure the images are opened as **composites** (without splitting channels). Otherwise, the script will quit with an error.
- During non-blocking dialogs, avoid closing any windows the algorithm does not explicitly ask you to close. The algorithm is designed to close most windows on its own and would run into error if it cannot close something that is expected to be open.
- In order to adjust the size of the ROI selection brush, you can use the main ImageJ control panel. Double click on the 'Oval' brush tool beneath the "Edit" menu to change the pixel size. ImageJ should save your preferred brush size between sessions.

Confirmation & Calibration

This pipeline combines Automatic Counting with the traditional pipeline of manually counting the cells of interest. It has two subdivisions, the Confirmation and the Calibration, which are independent, but work best when paired together.

Confirmation is the first pipeline that is triggered by ACCA and it is used to compare machine counts to human counts and relies on pre-processing and channel separation conducted by the automatic counter to prepare the ROIs to be reviewed. To this end, it always comes after automatic counting, there is no way to skip straight to confirmation as of the current version. Additionally, it offers a number of optional functions that can aid the comparison, discussed below.

Calibration pipeline, on the other hand, allows the user to calibrate various methods used for automatic counting. This is done through iteratively running the counter methods, prompting the user to fine-tune the parameters for each iteration, until a satisfactory set of parameters is generated. Then, this set is converted into a tuning profile code, which is printed in the log window, from where it can be retrieved by the user and manually inserted in the code (with a unique tuning parameter ID number). Please see methodTuning function for more information. The calibration pipeline is called from the confirmation pipeline, although the two are independent (so the user can run calibration without confirmation and vice versa).

Currently, the confirmation pipeline performs the following:

1) Close all windows and exit Batch Mode (otherwise, the user will not be able to see the images).
2) Open Auto-Counted image.
3) Open Original image. Optionally apply a user-defined pre-processing function (see the code for more information) to the image.
4) Open Cell Counter Plugin. This is a standard Fiji (a distribution of imageJ) plugin and

was not designed by the authors of this algorithm. Display the control panel, which allows the user to either move on to the next region or iteratively rerun the confirmation of the current region.

5) Optionally allow the user to overlay the automatic counts over their manual counter window.
6) Optionally allow the user to save their manual counter image. If the user chooses to save it, the image will also be used for the Calibration pipeline (if calibration of the region is enabled).
7) **If Calibration is enabled, call the calibration pipeline.** The confirmation pipeline does not need to be enabled for the calibration to start, although it is advised to have a manually counted (confirmed) image to aid in calibration.
8) Close all windows. Re-enable batch mode (if there are no more regions to confirm).

Some useful tips:

- During the manual confirmation, you can use any adjustments and plugins to alter the original (and automated) image. The algorithm is placed on hold until "Control Panel" dialog is triggered.
- In order to use the 3rd party Cell Counter plugin, first select the original image and then click "Initialize". Select any counter option (they differ only in color) and click on the cell of interest to place a count on top of it. Use the "Export Image" function to create an image with your counts permanently incorporated and then select this image if you enabled the saving of manual counter images. Otherwise, the algorithm will save the image without any counts on top of it.
- If the option to overlay automatic counts over manual counts is enabled, it is recommended to first do the manual counting and then overlay the automatic counts. This is because the automatic ROIs are drawn as bright spots to be easily recognizable and may interfere with visualization and manual counting.
- At the end of the pipeline, the script will ask the user to close any remaining windows. Please do not close any windows until then, as it may result in an error.

Currently, the Calibration pipeline performs the following:

1) Close all windows.
2) If the option to save manual counts is enabled, open the region manual counter image.
3) Open the original image for that staining channel and region.
4) Run the automated counter with the default parameters.
5) Display a control panel dialog that allows the user to input different parameters and re run the counter. This step is repeated iteratively, until the user is satisfied with the counter profile. To aid the visualization of the changes, the automated counts will be iteratively overlaid over the manual image (if one exists).

6) Print the last set of parameters in the Log window. It is formatted to work with no additional modifications when inserted in the methodTuning function.
7) Close all windows.

Some useful tips:

- Please consult the method-specific pages, described below, for a detailed overview of the role of each parameter in their respective method. You can also use the comments in code for the same purpose.
- When pasting the generated tuning profile, make sure it goes inside the "if" statement corresponding to a new Tuning Parameter ID (a number from 1 to Infinity). Then you will be able to specify this id for automatic counter moving forward. Currently the counter can accommodate 9 tuning profiles per stain, but this can be increased manually by changing the code.
- If both Confirmation and Calibration options are enabled for multiple regions, some menus from the Confirmation pipeline will transfer over to the Calibration pipeline, most notably the Cell Counter Plugin window. Please, avoid closing that window and just move on with the Calibration pipeline.
- During calibration, it may be helpful to select a single cell of interest and use ImageJ > Analyze > Measure to obtain the approximate values of minimum and maximum area of your cells of interest (Note, for 2D images Area = Volume). Additionally, ImageJ > Analyze > Histogram can be used to obtain the average thresholding value for your cells of interest. Tailoring these parameters to your cells of interest vastly improves the performance of the ACCA.

Methods

There are three important third-party methods utilized by the algorithm. Please follow the links below and study them before using the algorithm. Additionally, some of these methods require citation if you use this algorithm to analyze any manuscript data. Again, use the links below to find out how to properly cite these methods.

1) 3D Iterative Thresholding. Used in this algorithm to segment the cells of interest from the background and noise. https://imagejdocu.tudor.lu/plugin/segmentation/3d_spots_segmentation/start#d_iterative _thresholding ,
2) Particle Analyzer (comes with Fiji distribution of ImageJ). Used in this algorithm to count the segmented cells of interest. Also functions as a secondary filter to enforce circularity. https://imagej.nih.gov/ij/docs/menus/analyze.html#ap .
3) Cell Counter Plugin. Used in this algorithm as a tool to help the user manually count the cells during Confirmation. Please note, you can use a different tool if you want, in which case you do not need to cite or review this plugin. See the code for more information (line

1632 for ACCA V1). https://imagej.nih.gov/ij/plugins/cell-counter.html .

Code and Customizability

Structure

Currently, the algorithm is organized into 4 macro-blocks:

1) Initialization and main executable. Contains the user-defined variables, internal variables and the functionality to open the input database and loop over images.
2) processImage functional block. Contains the master-function that does automatic counting and calls Confirmation functions and all the minor functions (processRegion, countDapi, countCfos, countGFP, constructOverlap, countOverlap, enhanceRegion).
3) confirmImage functional block. Contains the confirmation master-function, which also calls calibration functions and all related minor functions (confirmRegion, overlayImage, manual pre-process functions)
4) calibrateMethod block. Contains master-calibration function and methodTuning database, which is essential for tuning the automatic counter. Also contains related minor functions (preserveCalibratedProfile, textConverter, roiReset, roiUpdate and prepareCalibrationOverlap).

Customization

Currently, the algorithm supports the processing of 4 ROIs per image, with 9 potential tuning profiles for each stain. The user can add more regions by editing the code responsible for setting up the processing of each region, which is mostly done by inputting and retrieving various variables, specified in processImage and confirmImage functions. In general, the user should not need to go any lower than these two functions. Please see the code for more information.

The user can also manually modify the tuneMethod function to add or alter the tuning profiles stored in that function. Please follow the general format and comments inside the function while carrying out any modifications.

User-Defined Variables:

Please see the first code block, titled User-Defined variables before running the algorithm. It is highly advised to set “triggerDialog” to 1 if it is the first time you are using the algorithm, as it allows the algorithm to guide user input with dialogs. Alternatively, you can set it to 0 and use all other user-defined variables to tune various parameters of the algorithm. Please see the comments in code to learn about the role of every user-defined variable.