User guide

Brightness Corrector plug-in for ImageJ

Sebastian Rassmann, Jan N. Hansen, and Jan F. Jikeli
research group Biophysical Imaging

Institute of Innate Immunity, Bonn, Germany

Funding: DFG priority program SPP 1926

Contact:

rassmann@uni-bonn.de

jan.hansen@caesar.de

http://www.iii.uni-bonn.de/wachten_lab/principal_investigator.html

Content

1 Introduction

1.1 Input images

The plug-in can handle single images, stacks and hyperstacks with a user-defined number of channels.

1.2 Mode of operation

BrightnessCorrector corrects intensity differences in an image or stack. To this end, an intensity map of the image is created. To generate the intensity map, the user selects one of the image's channels. Next, the map is created by normalizing each pixel's intensity in the image according to the intensity of the brightest pixel(s) within a user-defined radius. In the next step, the map is applied to the image. More specifically, each pixel value is divided by the corresponding map value, yielding the pre-processed image. Afterwards, the intensity values in the image are readjusted to match the new highest and lowest pixel value with the original image. Consequently, the values in between are adapted proportionally depending on their relative position within the range. Note, that the absolute range between images processed with different calculation settings may vary due to different maximum or minimum intensities in the pre-processed image.

To generate the map, a channel should be selected where (1) a positive signal can be detected across the entire image, and (2) the fluorescence of the labelled object is more or less equal across the image. For example, we use a DAPI staining to generate the map.

Figure 1 - Basic principle of the *BrightnessCorrector* plug-in: A map is created from the original image by normalizing each pixel's intensity based on the highest intensities within a used-defined radius. Then, the original image is divided by the created map and readjusted to the original range in pixel intensity resulting in a corrected image.

2 User Interface

2.1 General Settings Dialog

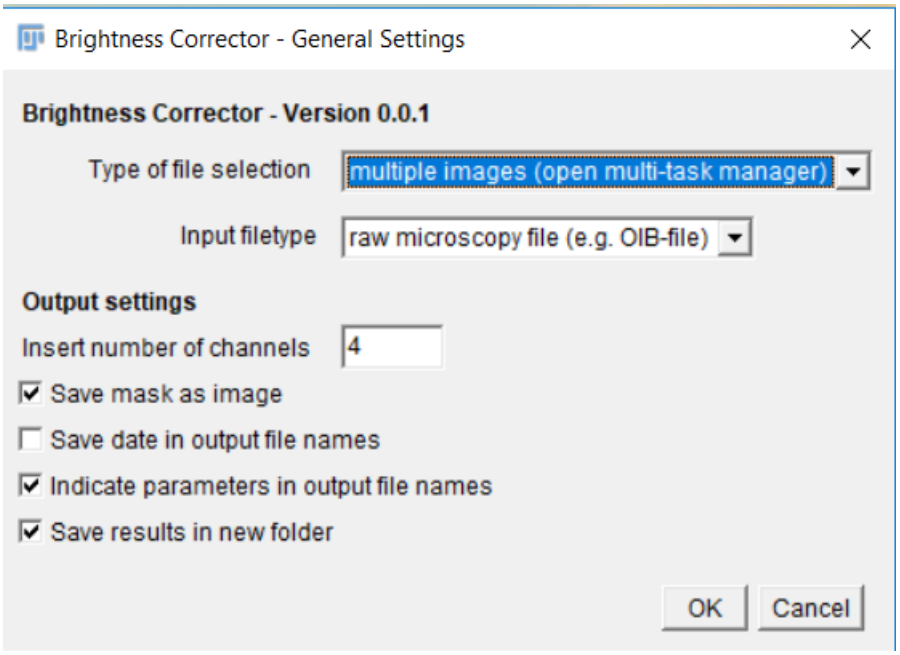


Figure 2 - First dialog for inserting general process settings

2.1.1.1 Type of file selection

This option lets the user choose how to insert the input images to the plug-in. Each transferred image will be processed individually.

- *active image in Fiji*

The active image in ImageJ is the image that is in the front (and does not show a grey blur over the image's name on top of the box around the image).

- *all images open in Fiji*

This setting will insert all images open in Fiji to the plug-in to be processed consecutively in the next step.

- *multiple images (open multi-task manager)*

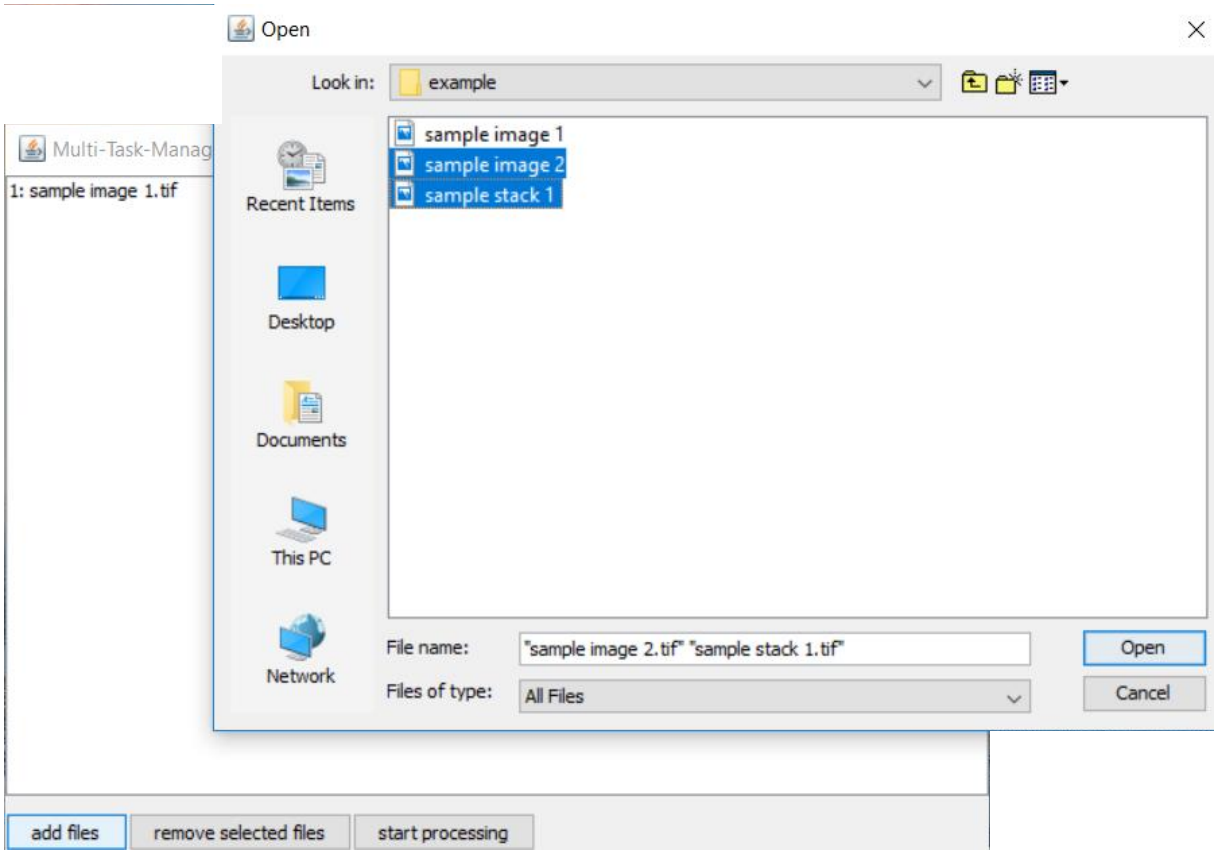


Figure 3 - Multi-Task-Manager and File Selector

This option provides the possibility of choosing multiple images from the hard drive. If selected, a second dialog appears (see Figure 3):

Click on "*add files"* to add new files. The appearing dialog enables the user to add files selected by a standard Java file selector. Click on "*Open"* to open new files. It is possible to open multiple files at once within the selector.

The names of the selected files will now appear in the window. They can be marked by clicking on the names and removed by clicking on "*remove selected files*".

"*Start processing*" will end this dialog and start processing of all images listed in the previous dialog.

2.1.1.2 Input file type

This option imports native microcopy files, if the *BioFormats* ImageJ plug-in is installed in the used ImageJ version. Thus, *BrightnessCorrector* processes raw images without a need for converting into TIFF format.

2.1.1.3 Insert number of channels

The user needs to specify the number of channels that each input image contains. Note, that it is not possible to process multiple images with a different number of channels at the same time. The plug-in will skip every image that does not contain the user-defined number of channels.

2.1.1.4 Save map as image

If this setting is selected, *BrightnessCorrector* will save the calculated map as an image to the same directory as the results.

2.1.1.5 Save date in output file names

If this setting is selected, *BrightnessCorrector* will add the date of processing to the file names of the output images. The format is yymmdd_hhmmss.

2.1.1.6 Indicate parameters in output file names

If this setting is selected, *BrightnessCorrector* will write the processing settings into the file names of the output images. This is useful if more than one setting for the same image is applied, which is commonly used to find the optimal settings to process a data set.

2.1.1.7 Save results in new folder

If this setting is selected, *BrightnessCorrector* will create a new directory for each image on a chosen location on the hard drive; a dialog (*choose Directory for results*) will be presented to the user to select a location. If this setting is not selected, all generated files will be saved in the same folder as the original image.

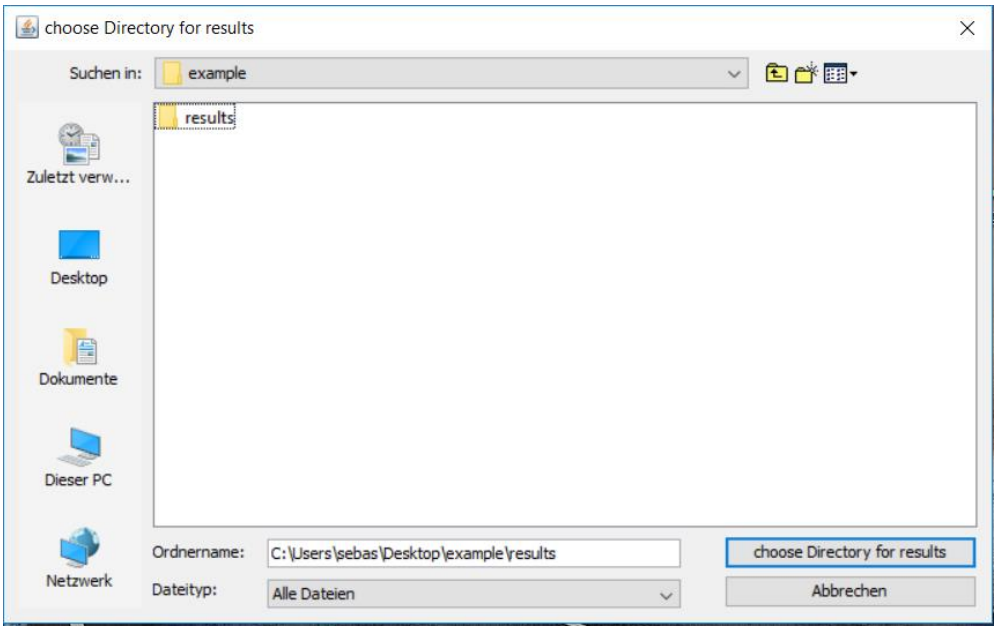


Figure 4 - Dialog for choosing the directory if files are to be saved to a new folder

2.2 Set image parameters Dialog

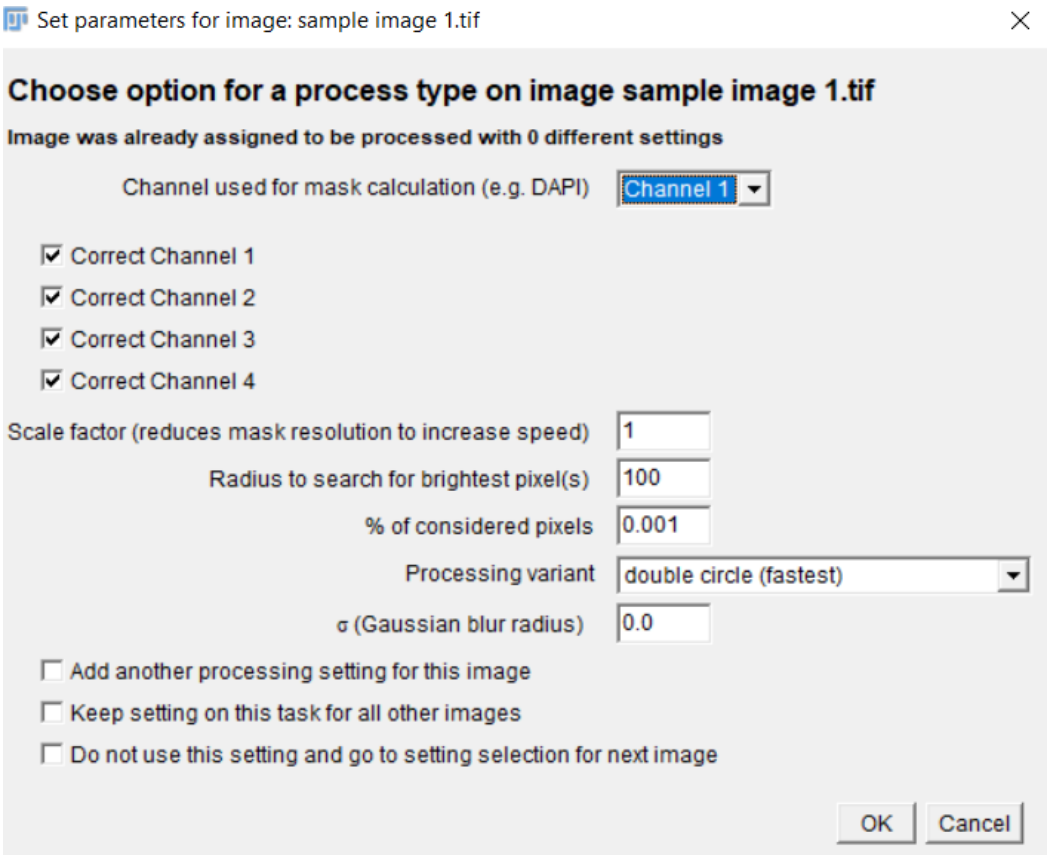


Figure 5 - Set image parameters dialog: This dialog allows the user to customize the process parameters. It will appear for each image unless the checkbox *Keep setting on this task for all other images* has been selected.

2.2.1 Set multiple process parameters

In the Dialog, the settings for processing each image can be individually chosen. This allows the user to process the same image with different settings to determine the best parameters. If the option "*save results in new folder*" is selected, all results are saved to one folder for each image.

The user can add another setting for the image by selecting "*Add another processing setting for this image"* and confirming with "*OK*". If this option is not set, the settings dialog will go on to the next image.

If "*keep setting in this task for all images"* is set, all chosen settings on the recent image will be applied to all following images.

With "*Do not use this setting and go to setting selection for next image*" the actual setting will be deleted and the dialog will proceed with the next image. Note, however, that it is not

possible to save the created settings for the following images - the settings needs to be set again in the following dialog.

2.2.2 Process settings

2.2.2.1 Channel used for map calculation

This option defines the channel used for the calculation of the map. The channel numbers are identical with those presented in the status bar of the image if opened in ImageJ.

2.2.2.2 Correct channel

These options define, which channels are corrected by *BrightnessCorrector*. Intensity values in channels, which are not selected, will be copied from the original image.

2.2.2.3 Scale factor *sf*

This option sets the scale of the map. If the set scale factor is different from 1, the map channel is copied and scaled according to the scale factor before it is used to determine the map. Both width and height are scaled according to the scale factor. Thus, the number of pixels in the map is reduced by the square of the factor. It is recommended to scale-down the map resolution, as the processing speed scales with the resolution of the map. Note, however, that this also blurs the image before determining the map, which can be beneficial to reduce noise (see Comparing the results for different settings, we conclude that using the *double circle* calculation and down-scaling the image is the best combination to improve performance. Here, the number of operations needed for the mask calculations is reduced by the square, but the area to be searched and the number of considered pixels is also reduced with the square. This decreases the runtime while minimizing the calculation error to invisibility.

Annex A - Results for a sample image).

2.2.2.4 Radius to search for brightest pixel(s) *r*

The radius *r* corresponds to the radius of the arbitrary circle, in which the plug-in will search for the highest pixel values to correct the pixel in the center of the circle. If the scale factor is set to >1, *BrightnessCorrector* will scale down both the radius and the map with the same factor, whereby the radius stays in proportion to the size of the map. Thus, the scale factor can be neglected for the insertion of the radius.

If generating the map based on an image of a DAPI staining, the radius should be chosen in a way that 5-20 cells will fit into a circle with the defined radius.

Note that the runtime of *BrightnessCorrector* depends on the circle area and thus, behaves proportionally to the square of the radius.

2.2.2.5 % of considered pixels *% used*

This value defines how many pixels as percent of the circle area (defined by the radius *r*) are averaged to generate the map. If *% of considered pixels* is set to 0.001, *BrightnessCorrector* will use only the brightest pixel - for all other set numbers, it calculates the number of considered pixels depending on the area.

It is recommended to set the number of considered pixels in a way that only foreground pixels are included at any position in the image. Thereby, it can be avoided that the relationship of fore- and background pixels in the considered pixels varies between different image regions, causing artefacts in the output image.

A higher number of considered pixels will result in a smoother map without any edges, better representing the intensity differences in the image and leading to better result. As a downside, the runtime increases exponentially with more pixels that need to be considered. To reduce runtime, downscale the image or add a Gaussian blur to smoothen the map instead of increasing the *% of considered pixels*.

2.2.2.6 Process variant

The setting determines the calculation method that is used for calculating the map. If set to *single pixel calculation*, *BrightnessCorrector* will use a native algorithm, resulting in a perfect map. If set to *double circle*, a heuristic approach is used to decrease the runtime (see Performance).

2.2.2.7 σ (Gaussian blur radius) *sigma*

This defines the standard deviation for the Gaussian blur applied to the mask. If set to 0, no blur is applied.

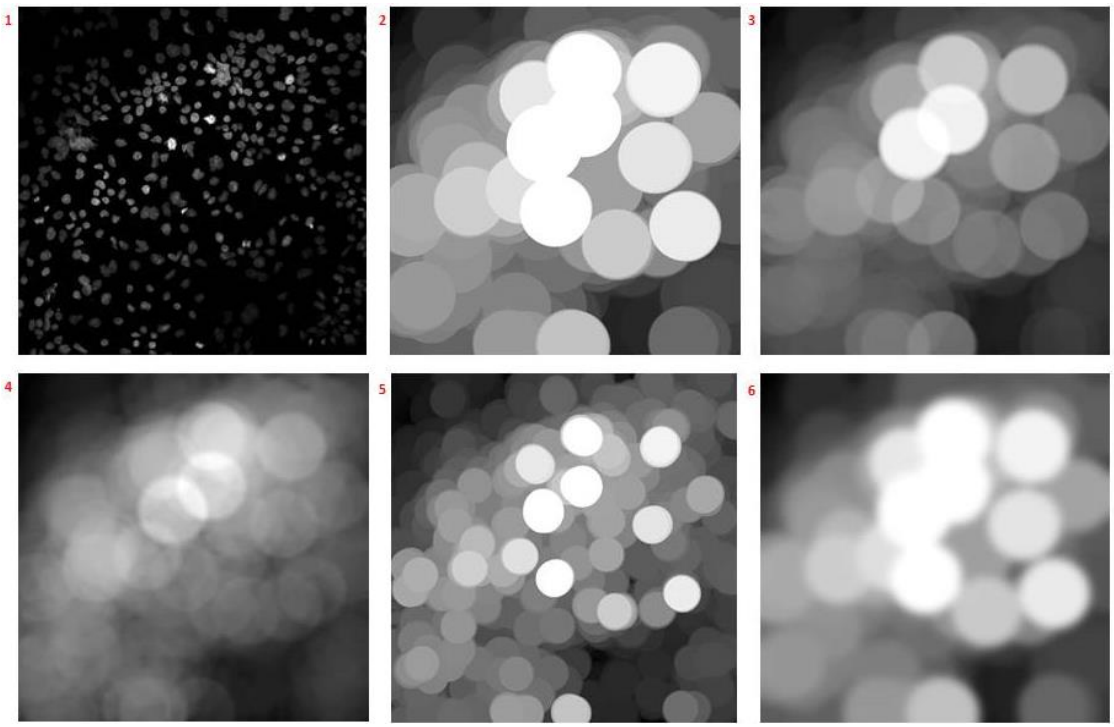


Figure 6 - masks created with different parameters on an image with a resolution of 1024*1024 pixels. 1: original image. 2: r = 100, single brightest pixel used (0.001% used). 3: r = 100, brightest percent used (1% used). 4: r = 100, brightest ten percent used (10% used). 5: r = 50, 0.001% used. 6: r = 100, Gaussian blur with σ = 14 applied.

2.3 Progress Dialog (*Multi-Task-Manager*)

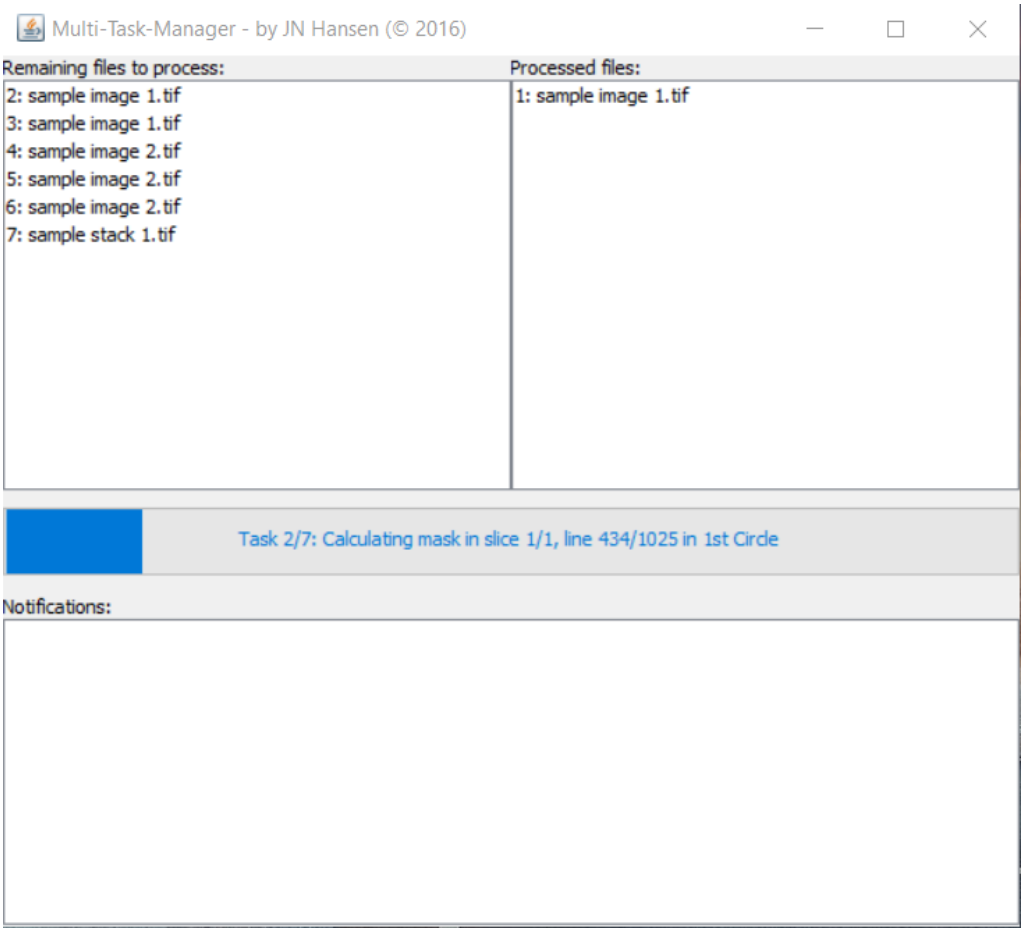


Figure 7 - Progress Dialog (Multi-Task-Manager)

The Progress Dialog Window appears during processing of the images and depicts the progress of the plug-in. Remaining files will appear on the left, processed files on the right. Note that if the same image was opened multiple times, the image will appear multiple times in the Task-Manager.

The Task-Manager will further notify about errors occurring during processing in the *Notifications*-panel.

2.4 Result

All output images are saved to the chosen directory (either same directory as the original file or a user-chosen one) and can be accessed directly after saving - there is no need to wait until all loaded files are processed. *BrightnessCorrector* will also create a metadata file containing all settings, the runtime and the date of processing. If the map was set to be saved, it will also appear in the same directory as the processed image.

3 Performance

3.1 Theoretical Background

Since images comprise big data sets, the runtime is an issue for the tool. For example, a radius of 100 pixels results in ~31,400 encircled pixels, which all need to be compared to examine the mean of the brightest pixels. This procedure is repeated for every single pixel in the image. For higher resolution images, not only the number of pixels that need to be corrected, but also the radius increases, because the same field of view is represented by more pixels. Thus, processing of large image files or stacks challenges even high-performance devices.

During processing, the circumferences of neighboring pixels consist of many identical pixels. Thus, applying this method independently to each pixel would be a highly-redundant operation, while slowing down processing.

Therefore, we developed the *double circle* algorithm, which creates a list (referred to as "Circle"-object in the source code) of only the brightest and hence relevant pixels in the perimeter of the pixel that is being processed. *BrightnessCorrector* moves over the image as depicted in Figure 8 while updating the list. Thus, the list is constantly adapted instead of calculated from scratch for every single pixel and the algorithm avoids redundant operations. Hence, the processing speed is increased (see For the runtime tests we processed five 16-bit images with a resolution of 1024 * 1024 pixels and 4 channels on a standard performing laptop. On high performance devices, runtime will be lower, albeit we expect the tendencies to be similar.

Runtime).

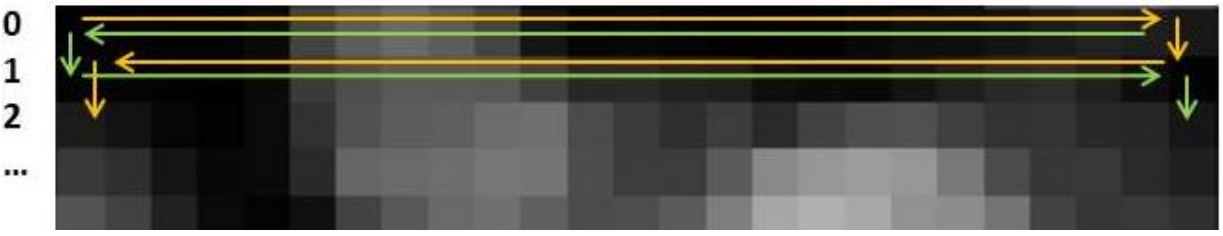


Figure 8 – processing scheme of *BrightnessCorrector*. The first circle (yellow) starts in the upper left corner, moves to the right, and returns in the opposite direction in the following line. The second circle (green) moves in the opposite direction of the first circle in every line.

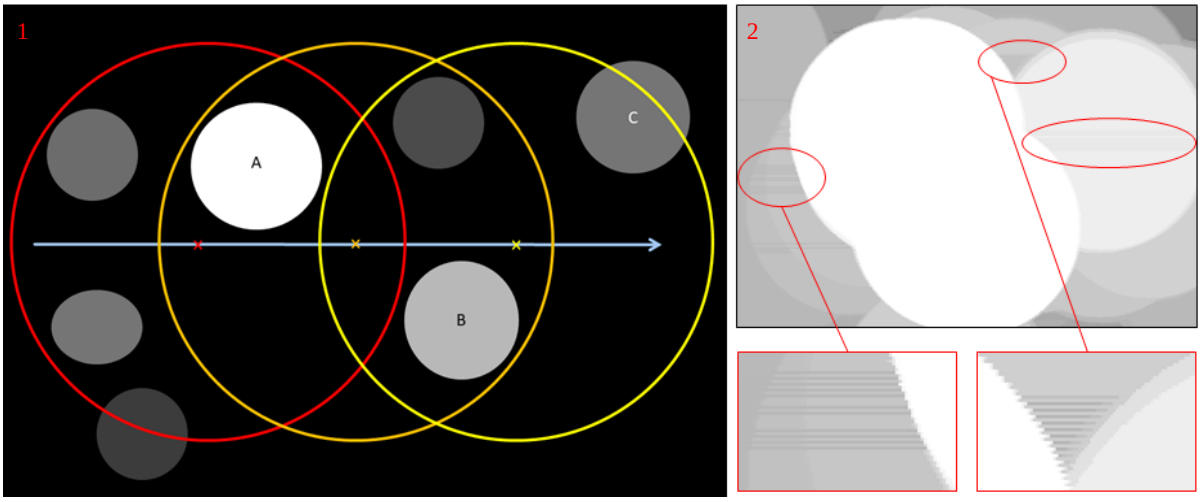


Figure 9 – An example, where the algorithm would be error-prone without using the second circle: 1: The circle moves over the image (blue arrow). First (red) the list of highest intensities is filled with A's high intensity pixels. Moving forward (orange), B's less intense pixels are prohibited from entering the list since A's intensities are higher than B's. As soon as A is outside of the radius (yellow) A's values are replaced by C's although B's would still be higher. This results in stripes every second line behind or in front of a bright object (inset). The stripes alternate from row to row as the calculation direction alternates, too (see also Figure 8 – processing scheme of *BrightnessCorrector*. The first circle (yellow) starts in the

One disadvantage of the *double circle* setting in *BrightnessCorrector* is the limited list size, hence it does not save all data (see Figure 9). While passing a bright object (A) containing a lot of bright pixels, the list is flooded with only A's pixels. This does not allow lower values from other objects (B) to enter the list. Moving forward, these missing values can result in errors in the map once A's values come out of range. Without B's values having entered the list, the algorithm cannot consider those values and calculates the map with darker values (C) instead. Since the list moves from one side to another, this occurs only after bypassing bright objects, e.g. for even-numbered lines on the right and for uneven-numbered lines on the left side of bright objects.

To minimize the error of not detecting some lower intensities following brighter intensities, the *double circle* algorithm moves another list (*2nd circle*) in opposite direction over the image in each line. The values calculated by moving the list in the opposite direction (*2nd circle*) overwrite the values calculated by moving in the original direction (*1st circle*), where the values of the *second circle* are higher.

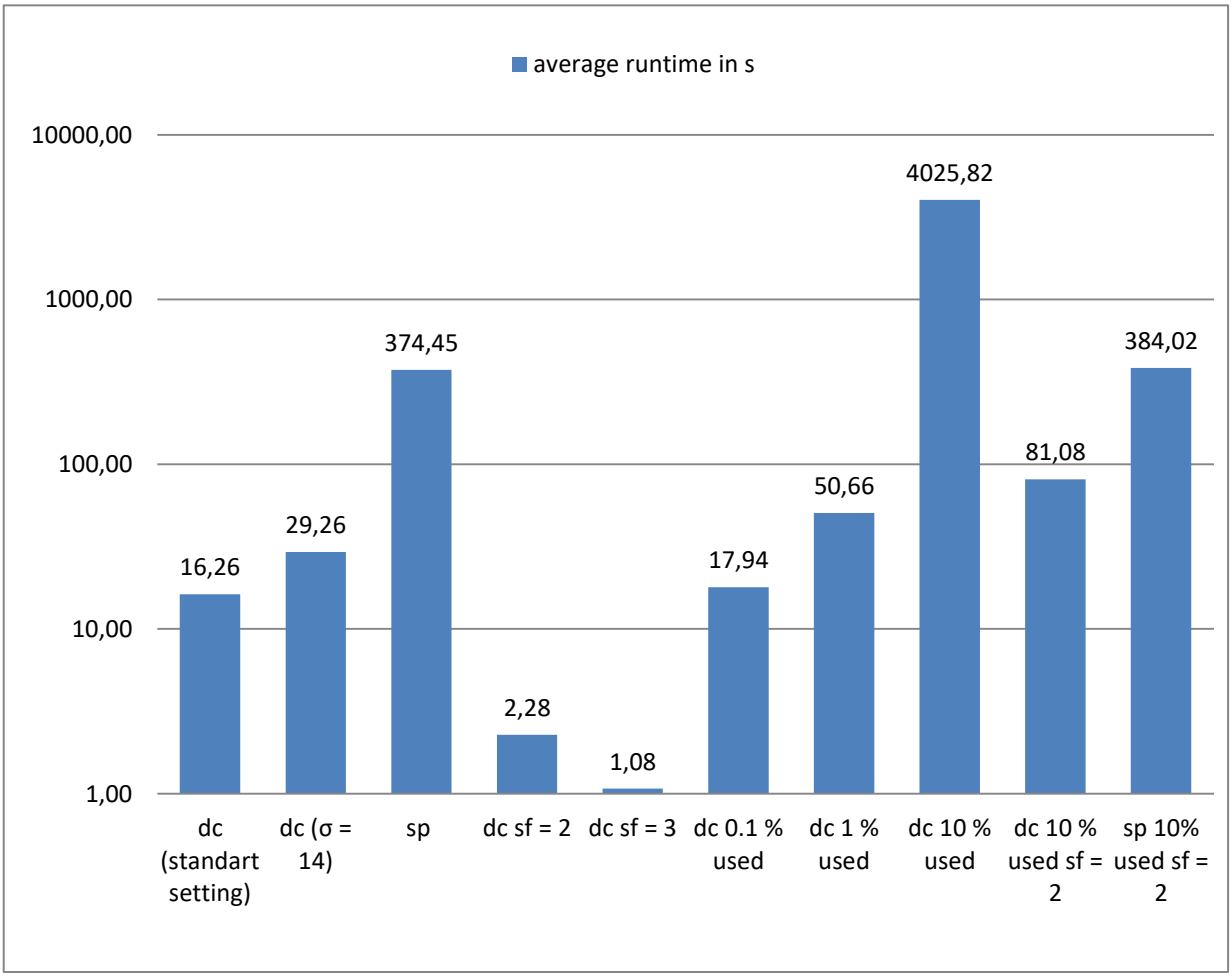
Still, due to minor errors in-between bright objects or in proximity to the edges of the image, the process is error-prone, so the results will slightly differ from the results of the native algorithm (see The plug-in's runtime highly depends on the chosen processing settings. On standard devices, the feasibility of applying some settings is restricted by the runtime. E.g. using *single pixel calculation*, a high number of considered pixels, or a large image can highly increase the runtime to infeasibility.

Quality). Finally, the user can decide whether the process shall be performed according to high quality or shorter runtime by applying the native algorithm (*single pixel calculation*) or the runtime-optimized one (*double circle*), respectively.

3.2 Performance examinations

For the runtime tests we processed five 16-bit images with a resolution of 1024 * 1024 pixels and 4 channels on a standard performing laptop.[1] On high performance devices, runtime will be lower, albeit we expect the tendencies to be similar.

3.2.1 Runtime



The plug-in's runtime highly depends on the chosen processing settings. On standard devices, the feasibility of applying some settings is restricted by the runtime. E.g. using *single pixel calculation*, a high number of considered pixels, or a large image can highly increase the runtime to infeasibility.

[1] Configuration: Intel Core i5-6200U with integrated HD Graphics 520 and 8 GB of RAM on Windows 10 (64 bit)

3.2.2 Quality

3.2.2.1 Single Pixel Calculation vs. Double Circle Calculation

To demonstrate the advantages and disadvantages of the calculation methods, we tested *BrightnessCorrector* in both calculation methods and compared runtime and the resulting images.

Processing the sample images with the single-brightest-pixel method (*0.001% used*) yielded an average difference in pixel intensity of 0.488 for comparison of the *single pixel* calculation (perfect calculation) to the *double circle* calculation (fastest runtime) (

Table 1). For a standard camera range of 12-bit, this corresponds to an error of 0.0119%.

image number	1	2	3	4	5	Average error	% (of 4096)
dc, sf = 1 vs. sf = 2, 10% used, output image	1.695	1.614	1.403	13.567	7.650	5.186	0.126606%
dc vs. sp, sf = 2, 10% used, output image	0.310	0.292	0.205	0.603	6.649	1.612 (excluding image 5: 0.353)	0.039351% (excluding image 5: 0.008606%)
dc vs. sp, sf = 2, 10% used, map	1.039	1.068	0.622	0.224	0.550	0.701	0.017104%

Due to long runtimes it was not feasible to calculate the map using the single pixel calculation when using a higher percentage of the circle area for the calculation (10% used). Thus, we used the scale-factor option (sf=2) for the quality comparison. Yet, the average difference in pixel intensities between the *single-pixel* calculation and the *double-circle* calculation was 1.61 (0.0394%) in output images, while 0.70 (0.0171%) on the map. This high error rate was a consequence of one outlier (image 5, Table 2), in which the intrinsic error of the double-circle calculation was amplified by readjusting the intensity range after correcting the image. This amplification appears, when minimum or maximum values of the pre-processed image are affected by the intrinsic error of the *double circle* calculation (see Mode of operation). Excluding this image for the error calculation, the error would only be 0.35 (0.00861%).

3.2.2.2 Effects of downscaling

The calculated difference between scaling (sf = 2) and not scaling the image (sf = 1) for 10% *considered pixels* resulted in an average error on the output image of 5.19 (0.127%) (see Table 2). However, in the fourth image (Table 2) the intrinsic errors of the *double circle*

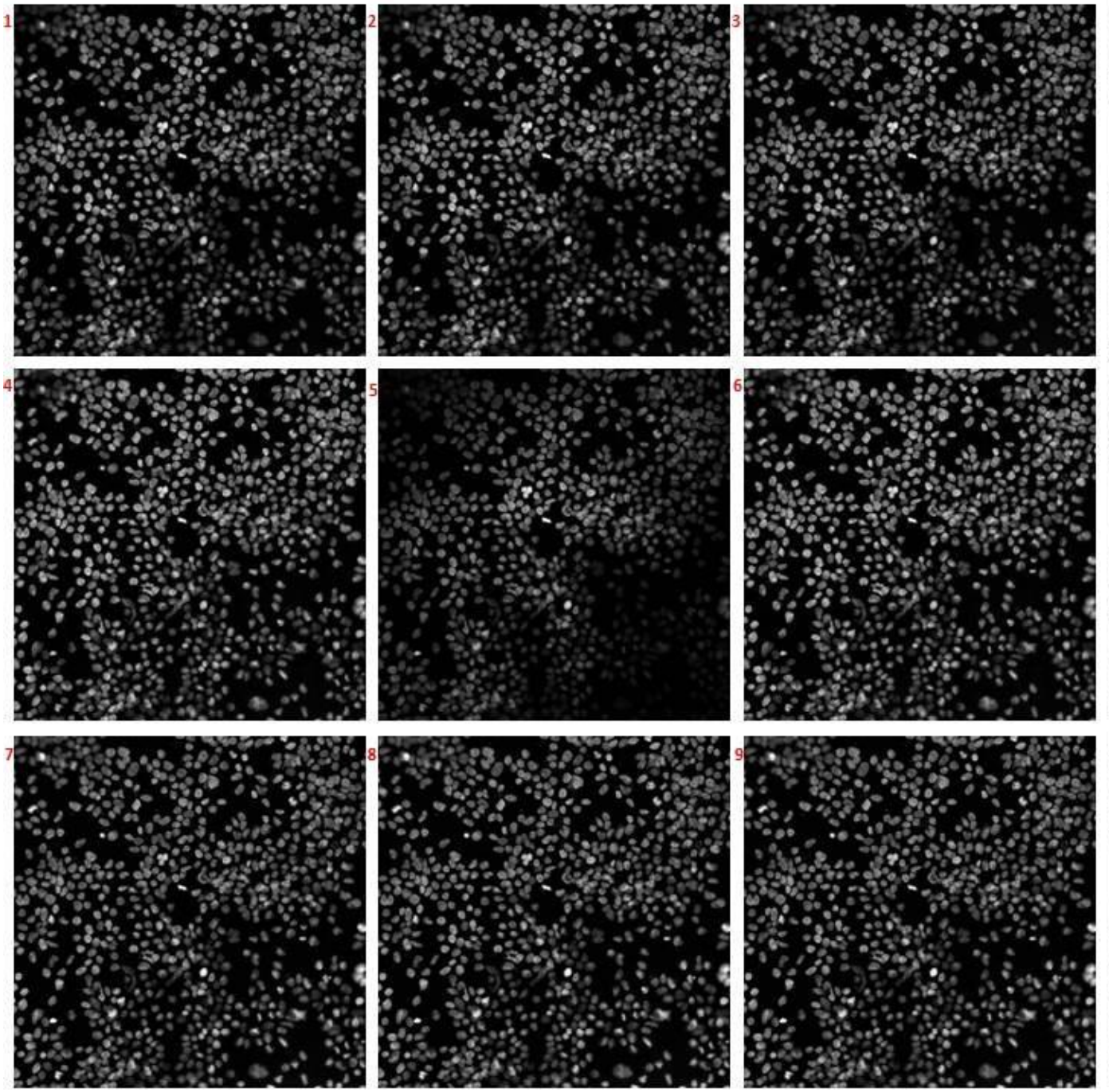
algorithm also appear to have affected the minimum or maximum value and, hence, different intensity adjustment in each of the compared images resulted in a higher difference in intensity values (see 3.2.2.1). Here, the maps have different sizes and a comparison is not possible.

3.3 Discussion

Comparing the results for different settings, we conclude that using the *double circle* calculation and down-scaling the image is the best combination to improve performance. Here, the number of operations needed for the mask calculations is reduced by the square, but the area to be searched and the number of considered pixels is also reduced with the square.[2] This decreases the runtime while minimizing the calculation error to invisibility.

[2] Hence fewer list operation and sorting has be done which further speeds the process up so performance in the test increases even further than the expected $cf^2 * cf^2 * cf^2 = cf^6$. Since not only the mask calculation but also the image calculation takes some time performance does not increase a lot by increasing sf from 2 to 3.

4 Annex A - Results for a sample image



Comparison of different settings; if not differently declared, default settings were applied (sf = 1, r=100, 0.001% used, σ=0). 1: Double Circle, 2: Single Pixel Calculation, 3: Double Circle (sigma = 14), 4: Double Circle (1% used), 5: Unprocessed Image, 6: Double Circle (r=50), 7: Double Circle (10% used), 8: Double Circle (sf=2, 10% used), 9: Single Pixel Calculation (sf=2, 10% used).

Since the overall intensity level of the image differed between the different processing parameters (see I staining to generate the map.), image 4's brightness has been increased by factor 1.4 and images 7-9s' by factor 1.7 for better visualization.

5 Annex B - Error Calculation

Table 1 - Errors in down-scaled setting

image number	1	2	3	4	5	Average error	% (of 4096)
dc, sf = 1 vs. sf = 2, 10% used, output image	1.695	1.614	1.403	13.567	7.650	5.186	0.126606%
dc vs. sp, sf = 2, 10% used, output image	0.310	0.292	0.205	0.603	6.649	1.612 (excluding image 5: 0.353)	0.039351% (excluding image 5: 0.008606%)
dc vs. sp, sf = 2, 10% used, map	1.039	1.068	0.622	0.224	0.550	0.701	0.017104%

Table 2 - Errors between native and Double Circle algorithm

image #	1	2	3	4	5	Average error	% (of 4096)
mask average	4.333	4.192	2.913	2.069	0.547	2.811	0.068623%
image average	0.709	0.522	0.635	0.439	0.137	0.488	0.011924%