POLYTECHNIQUE MONTRÉAL

UNIVERSITÉ D'INGÉNIERIE

User guide for:

SAMScratch

Systematic Area Measurement of Scratch

Developed by :
Fiona Milano & Laura Ahunon

Members of the Biomaterial and Cartilage Laboratory, a research group of Polytechnique Montréal

Version 1 – December 2020

Table des matières

General information

SAMScratch is a MATLAB standalone application autonomously detecting scratches on scratch essay images and computing the scratch area. This software allows for a repeatable and easy measurement of the scratch area. When analysing a time series, it also computes the migration speed. The numeric results obtained with the application can be save either in excel or text format. The application shows you the scratch edges detected by superimposing them on your images (Figure 1). This figure can be saved as well as the linear fit of your data when analysing a time series (Figure 2).

This software is freely available for both MAC and WINDOWS at https://github.com/Biomaterials-and-Cartilage-Laboratory/SAM-Scratch. The application requires the MATLAB component runtime (MCR) version 2019b to run, which takes ~2 GB storage. This version of the runtime is freely available at https://www.mathworks.com/products/compiler/matlab-runtime.html. To install the software, please refer to the "installation" section. The MATLAB code of the algorithm behind SAMScratch is also given in our Github.

The present document describes the algorithm behind SAMScratch, details the installation procedures for MAC and WINDOWS and explains how to use the interface. If you need any help to use the software or if you have ideas to improve it, feel free to contact us (fiona.milano@polymtl.ca or laura.ahunon@polymtl.ca).

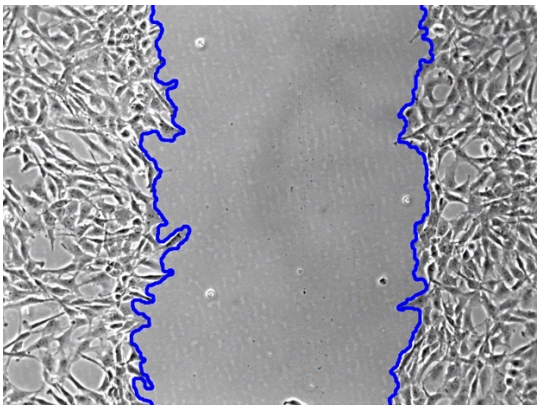

Figure 1. Scratch edges detected by SAMScratch superimposed on the original image.

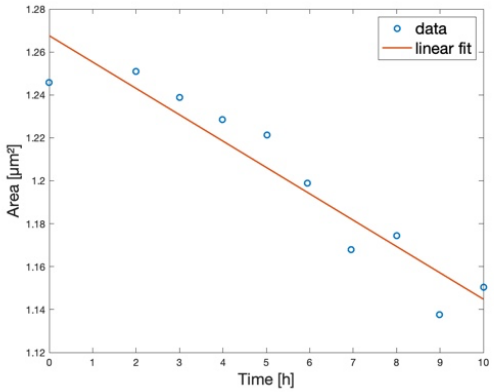

Figure 2. Linear fit of the scratch area evolution in time obtained when analyzing a time serie. The migration speed is the slope of the linear fit (here 4.43μm/s). SAMScratch also give the line equation and the coefficient of determination.

Legal information

By downloading and/or installing the SAMScratch software, referred to below as the SOFTWARE, you agree to the following license agreement.

- The Biomaterial and Cartilage Laboratory group of Polytechnique Montréal, referred to below as BCL, agrees that the LICENSEE uses the SOFTWARE for *non-profit* and *non-commercial* purposes only. SOFTWARE shall mean all SOFTWARE provided by BCL, including any derivatives thereof. The software is intellectual property of BCL and all rights, title and interest, including copyright, remain with BCL.
- Nothing in this agreement shall be construed as granting any license under any intellectual property right vested to BCL or any right to use the SOFTWARE or any BCL information other than herein expressly specified.
- In any publications resulting from research done under the use of the SOFTWARE, the LICENSEE will properly cite either party's contribution. Any report or publication of results obtained with the SOFTWARE will acknowledge its use by stating that SOFTWARE « was developed by the Biomaterial and Cartilage Laboratory (BCL), at Polytechnique Montréal ». Any published work that utilizes the SOFTWARE shall include the following reference:
    - Ahunon, L., Milano, F., Chevrier, A., & Lavertu, M. (2020). A novel image analysis algorithm reveals that media conditioned with chitosan and platelet-rich plasma biomaterial dose dependently increases fibroblast migration in a scratch assay. *Biomedical Physics & Engineering Express*, 2020, *6*(6), 065021.
- Proper citation is essential to continued BCL funding for research, as it is the primary way in which we demonstrate the value of our research to the scientific community. Thank you for your attention to these details.
- The BCL does not assume any responsibility for the SOFTWARE and the use of it. BCL will not assume any liability for damages or loss of data occurring through the use of the SOFTWARE. BCL does not guarantee the suitability of the SOFTWARE for any application. The LICENSEE will hold BCL harmless for any claims on damages or loss of data which occur during the LICENSEE's use of the SOFTWARE.
- BCL does not guarantee that the SOFTWARE does not infringe any third party's intellectual property rights (including copyrights, patents, patent applications, publications or any other type of intellectual property right). The LICENSEE has to acquire on his/her own all necessary licenses and permissions for the use of the SOFTWARE if not otherwise agreed in writing.
- In case the SOFTWARE is or will be under the physical control of the LICENSEE before this agreement is signed, BCL gives consent to the use of the SOFTWARE under the condition of the LICENSEE's prior consent to this agreement.
- The use and distribution of the MATLAB Compiler Runtime (MCR) libraries is subject to the MATLAB license agreement (see MCR_license.txt). In particular, the MCR may not be redistributed in any way separate from the SOFTWARE.

Algorithm description

With most microscopes, the obtained images belong to the RGB color space. They are made of a two-dimensional array of pixels. Each pixel is defined by an integer triplet. Each triplet contains the red, green and blue proportion making the color of the corresponding pixel. The SAMScratch algorithm starts with the extraction of one of these three channels, leading to the obtention of a grayscale image as presented in Figure 3a. The green channel is chosen as default because for most microscope images it shows the highest contrast. This choice can be modified in the application (see Interface use and parameters and options sections).

Once a grayscale image is obtained, the gradient of this image can be computed. The gradient represents the change in the intensities of the image (when neighbor pixels are compared): it is higher in regions of the images showing high variability in intensities (e.g. edges) but null in regions made of a single intensity (uniform regions). Figure 3b shows the gradient of the grayscale image. Computing the gradient of an image is therefore a way to highlight the edges of objects in this image. Light pixels indicate higher values of the gradient while dark pixels correspond to lower values. However, the gradient computation uses derivatives while images are made of discrete points, allowing only an approximation of the gradient. Several operators can be used to perform such an approximation. Sobel's operator is used here as default, but this parameter can be modified (see Interface use and parameters and options sections).

The edges of the cells in the images should now be extracted so that the cell sheet can be identified. To extract the cells edges, the gradient should be thresholded to keep only the highest gradient values corresponding to real edges and getting rid of the noise. This threshold is heuristically determined by MATLAB but then artificially lowered, allowing to extract not only the cells edges but also their content, corresponding to lower magnitude of the gradient (Figure 3c). Extracting also the content will ease the identification of the cell sheet. Artificially lowering the heuristic threshold is done by multiplying it by a number between 0 and 1. This number is here called the thresholding factor (TF). It is the main parameter influencing the final result (scratch identification), as discussed in the parameters and options section. TF was empirically set to 0.6 as a default value.

Once the cells' edges and content have been identified, the cell sheet is reconstructed by an opening operation. The opening operation allows to suppress the white noise present in an image. It is based on the use of a "structuring element". The structuring element is described by a shape and a size (e.g. a disk with a radius of 10 pixels). The opening operation allows to suppress all white objects in the initial image which cannot contain the structuring element. By removing small white objects, this operation allows to reconstruct the cell sheet, as seen in Figure 3d. The choice of the default structuring element shape and size is discussed in the parameters and options section.

Once the cell sheet is reconstructed, a closing operation is used to clean its borders. Closing is the complementary operation to opening it removes small black objects from an image, as seen in Figure 3e. This operation is also discussed on the parameters and options section.

The image is then further cleaned by removing every black object which is not connected to the image border. As the cell sheet will always be connected to the borders, this allows to remove every other black object contained in the image, which correspond to noise. This operation is shown in Figure 3f. Following the same idea, knowing that the scratch is also linked to the image borders, every white object which is not connected to the border can be removed as it likely corresponds to noise in the cell sheet. This is illustrated in Figure 3g. Due to this step of the algorithm, SAMScratch only allows to detect scratches which are linked to at least one border of the analyzed image. This choice was made as most scratch essays end at 50% scratch closure, when the scratch is not yet closed.

Finally, to detect the scratch, every remaining white object is identified, and their respective area is computed (Figure 3h). The scratch is identified as the largest of those objects (Figure 3i).

The app automatically segments the image by using the default value for every parameter. If the user is not pleased with the obtained result, they can adjust all parameters as explained in the parameters and options section (TF, gradient type, color channel, shape and size of the two structuring elements).

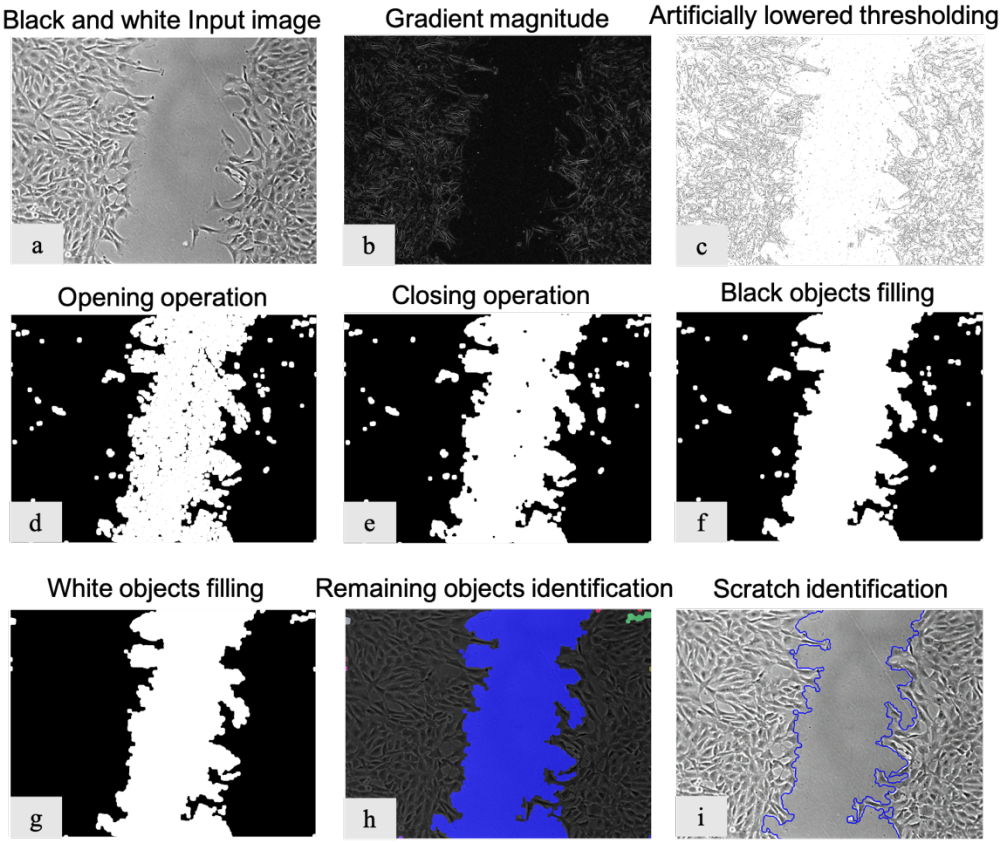

Figure 3. Flowchart of the SAMScratch algorithm for scratch detection and area determination. a) Input grayscale image or selection of a color channel. b) Computation of the gradient to find edges and cells content. c) Thresholding of the gradient (threshold heuristically computed by MATLAB then artificially lowered). d) Reconstruction of the cell sheet by using an opening operation. e) Reconstruction of a detailed sheet border by using a closing operation. f) Filling black objects to remove black noise in the scratch area. g) Filling of white o jects to remove white noise in the cell sheet area. h) Identification of the remaining white objects. i) The biggest object is the scratch.

Installation

Installation on WINDOWS is straightforward but installation on MAC is more complicated. If you have access to a WINDOWS computer, we therefore recommend you use it rather than a MAC.

Mac

Explanation to come.

Windows

To install the SAMScratch app on WINDOWS, please follow the instructions below:

1. Download the file named `SAMScratch_Windows` from GitHub (https://github.com/Biomaterials-and-Cartilage-Laboratory/SAM-Scratch).
2. A message stating "SAMScratch_Windows is not a commonly downloaded file. It could be dangerous" could appear. Do not take it into account and go on with the download.
3. Unzip the file `SAM-Scratch-master.zip` and extract SAMScratch_Windows.exe.
4. Double-click on the file `SAMScratch_Windows.exe` to unpack the program files.
5. Upon unpacking, the runtime should be automatically installed (Matlab Component Runtime – MCR – 2019b). The runtime is necessary to run the app and will take ~2 GB storage (and therefore require quite some time to get downloaded). If the runtime is already installed on your device, it should be detected by the app and not trigger another downloading. If the installation is not successful, you can install the runtime manually from https://www.mathworks.com/products/compiler/matlab-runtime.html (version r2019b).
6. Once the runtime is installed, you can start the app by clicking the `SAMScratch_Windows.exe` file. You can add shortcut for the app on your desktop (by checking the appropriate option at the end of the unpacking or manually later) and use the images given in `icon.zip` folder provided on the GitHub as an icon for the app if you like.

Interface use

When launching the SAMScratch application, a window similar to Figure 4 appears. SAMScratch supports all the graphic files formats supported by MATLAB: JPG, PNG, TIFF, TIF, BMP, GIF, HGF, PCX, XWD and others.

SAMScratch allows you to analyze unrelated images, or images issued from a time series (same scratch analyzed through time to compute a migration speed). After the images acquisition, if you wish to analyze a time series, put all the images of a single time series in the same folder. For the program to be able to analyze the time series, name the images as `name_time.extension` where `name` can be anything but `time` is the time of the image acquisition in minutes. They must be separated by an underscore and time must be given by a number only (e.g. `Plate-1-Well-B4_120.tif` for an image taken after 2 hours, 120minutes are indicated). **Make sure not to use an underscore anywhere else in**

the name of the file and think this through prior to your experiment to save time during the analysis. If any image contained in the folder is named otherwise, the app will let you know through an error message. If you do not wish to analyze a time series, your images can be identified by any name you want.

Once the application is launched, image(s) can be analyzed through the following steps:

1. First, in the ***images directory*** panel, select the type of analysis you wish to perform (number 1 in Figure 4). It can either be a single image, a folder containing several images to be analyzed separately or a folder containing a time series (i.e. images from which you want to compute a migration speed).
2. Click the ***Browse…*** button to select a directory (an image or a folder depending on the type of analysis).
3. Launch the analysis by clicking on the button ***Go***. The program will then analyze the selected image, or all the images contained on the selected file. The segmented image(s) will be displayed as shown in Figure 5, with the computed scratch area above each image. If you are analyzing a time series, a window with a linear fit applied to the area measured will be displayed, as well as the computed migration speed and the regression coefficient R^2 (see Figure 6). The default unit of the results is in % of the total image area for the scratch area and in pixels/hours for the migration speed. To allow for an area in mm^2 and a speed in μm/h, you must select μm as a unit in the control panel (number 2 in Figure 4) and enter the number of pixels corresponding to 1 μm for the analyzed images in the window that appears.
4. When analyzing more than 4 images, Click the "<" and ">" buttons to browse through the analyzed images. The numbers next to the images show the image number. You can also click ">>" and "<<" to move to the end or the beginning of the dataset (number 3 in Figure 4).
5. If the algorithm fails to automatically find a satisfying result for some images, the settings be modified by the user to obtain a more plausible one. The simplest way is to modify the thresholding factor (TF). This can be done for a specific image through the tools on the left of the image, either by choosing a value through the slider (number 4a in Figure 4) or by entering a value in the box below the slider (number 4b in Figure 4). The default threshold factor value is of 0.6. If you wish to change this value for every image analyzed, this can be done by modifying the TF value in the control panel and clicking the ***apply to all images*** button (number 4c in Figure 4). This applies the same chosen TF to every image. If you wish to apply different thresholds to different images, it has to be done separately for every image. If a satisfactory result cannot be obtained simply by changing the threshold factor, more parameters of the algorithm can be changed by clicking on the "detail" button on the left of an image (number 5 in Figure 4). This will open a new window. The parameters accessible through this window will be discussed in the parameter and options section. In the case of a time series, if you change the segmentation of some images, you must click on the ***update*** button in the migration speed window to take your changes into account in the computation of the migration speed. Alternatively, you can hit the ***Migration speed*** button in the main window (see Figure 5, below the images directory panel, this button only appears when you analyze a time series).

6. If you are less satisfied after changing parameters than with the initial result given by the algorithm, you can reset the analysis in the control panel (number 6 in Figure 4).
7. Once you are satisfied with the obtained results, you can click on the ***save results*** button (number 7 in Figure 4). You will then be able to store your results in a *.txt* or *.xlsx* file. The saved file will contain the name of the analyzed image the area measured by the algorithm, the value of each parameter used to compute the obtained result and, in the case of a time series, the migration speed, the R^2 and the equation of the linear fit. See the output files section for more details.
8. Your original images with the segmented scratch superimposed can also be saved by clicking on the ***save images*** button (number 8 in Figure 4). Similarly, you can also save the linear fit graph of your data in the case of a time series by clicking on the ***save graph*** button in the migration speed window.
9. When your analysis is completed and your results are saved, you can perform a new analysis by clicking on the ***new analysis*** button in the control panel (number 9 in Figure 4).

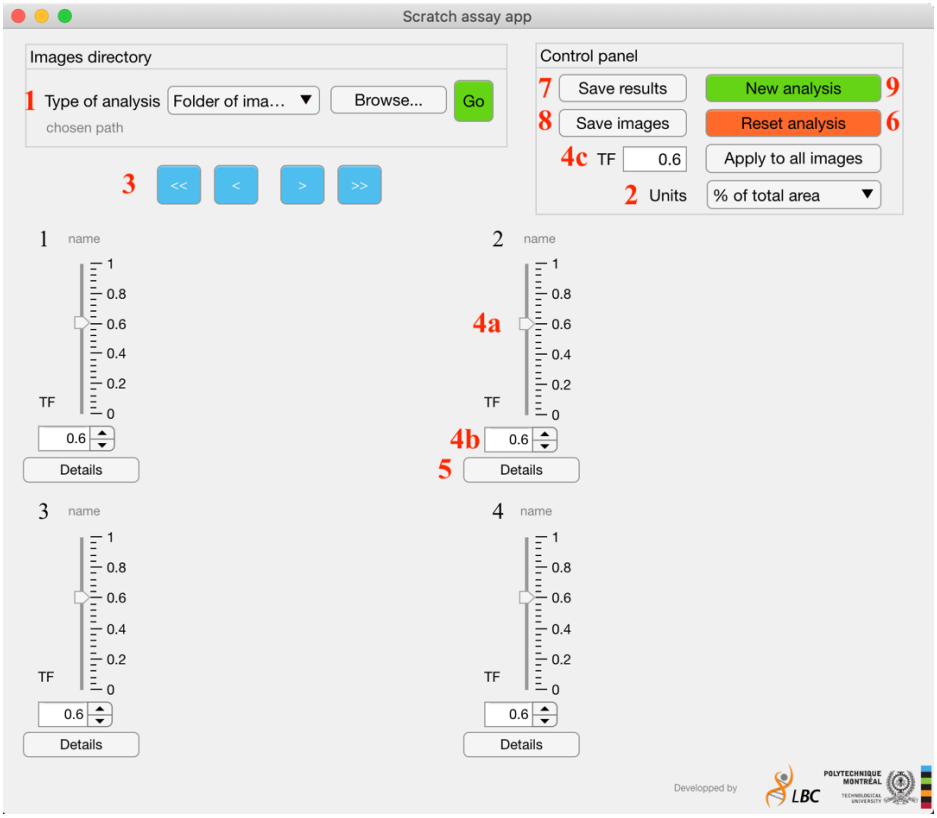

Figure 4. SAMScratch window after launching the application

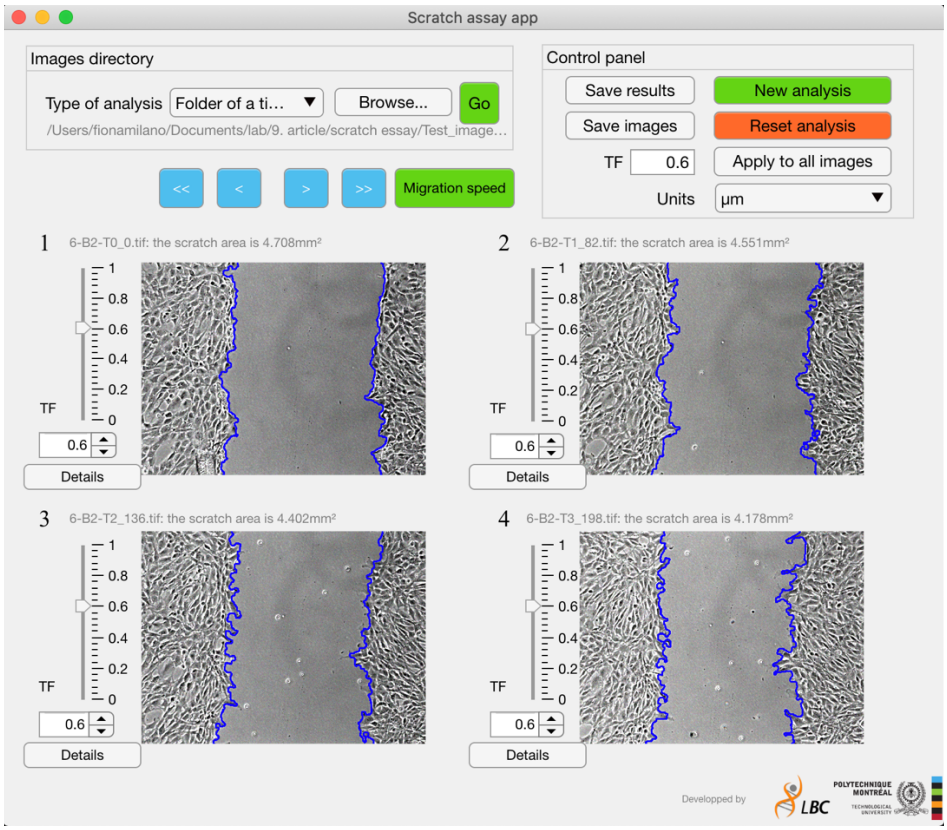

Figure 5. SAMScratch window after launching the analysis

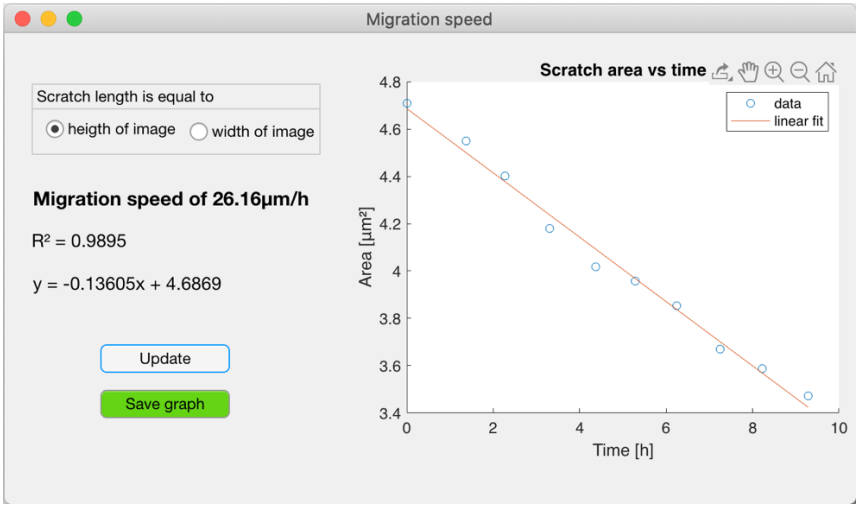

Figure 6. Computation and display of the migration speed in the case of a time serie.

Parameters and options

When clicking on the ***details*** button (number 5 in Figure 4), the ***detailed changes*** window appears as in Figure 7. This window allows you to change most of the algorithm parameters, and so to fine tune the result obtained. When changing a parameter, the

modification triggered in the scratch segmentation is directly shown on the scratch image. However, these changes are not saved automatically. If you are not pleased with the result and prefer the automatically obtained result, you can either click ***reset parameters*** and try some other changes, or close the window without saving the changes and return to the main app. If you are pleased with the result, you can apply it to the current image you are working on (***apply changes to current image*** button), or even apply it to all images analyzed (***apply changes to all images*** button).

The parameters you can modify through the *detailed changes* window are hereafter detailed. To understand which role these parameters play in the whole algorithm please refer to the Algorithm description section. Here, you will only be given a hint of the influences of the parameter on the final result. We suggest you start by modifying the ***thresholding factor***, already available in the main app, before touching any other parameter. The reason being that the thresholding factor is the parameter having the biggest effect on the final result and is the easiest to fine tune. The available parameters are the following (from left to right and up to bottom in the control panel of Figure 7):

The gradient detection method

The *detailed changes* window allows you to choose between 6 gradient detection methods:

- *Sobel's filter*. It detects preferentially vertical and horizontal lines and is computationally not expensive. By default, Sobel's operator is here used as the cells are usually spread along the horizontal or the vertical axe of the image. If the cells are oriented along a different axis you can try to change the gradient detection method to improve the segmentation result. The others possible methods are:
- *Prewitt filter*. It also detects preferentially vertical and horizontal lines.
- *Roberts filters*. It detects preferentially diagonal lines. It is therefore interesting if cells are oriented along the image's diagonal. It is also not computationally expensive, however is more sensible to noise in the image than Sobel and Prewitt.
- *Log filter*. It filters the image with a Laplacian of a Gaussian (LoG) filter then finds edges by looking for zero-crossing.
- *Canny filter*. It computes the image's gradient by using the derivative of a Gaussian filter. The Canny method then uses two thresholds to detect strong and weak edges from the gradient. Using two thresholds makes this method less sensible to noise but more computationally expensive. It is therefore interesting when you image contains noise.
- *Approxcanny filter*. It is an approximate version of the Canny filter, allowing a faster execution time. However, it is less precise.

The color channels

With most microscopes, the obtained images belong to the RGB color space. Our algorithm works with grayscale images. It therefore starts with the extraction of one of the three channels to obtain a greyscale image. The green channel is chosen as default as it shows the highest contrast for most microscope images. However, it is possible in some

situations that one of the two other channels will show more contrast. This will typically be the case when studying fluorescence images.

Opening element

After gradient computation and thresholding, the cell sheet is reconstructed by an opening operation. Its effect is to preserve white objects which have a shape similar to the one of the structuring element, or which can completely contain it. The thresholding operation allows it to reconstruct the cells edges. We therefore want to use an opening element with a size similar to the one of the cells, as it is small enough to preserve the scratch but big enough to erase the cell interior. By default, a disk element with a radius of 10 pixels is used (approximately half the size of the cells). The disk is used as it is close to the shape of the cells. If your cells are really asymmetric, you can also try a rectangle structuring element. If cells are bigger in your images and the cell sheet is not completely reconstructed, try to increase the structuring element size.

The structuring element shape proposed by the SAMScratch app are:

- *Disk*. The size of the element corresponds to its radius.
- *Rectangle*. Two dimensions must be given by the user: the width and the height of the rectangle. When switching from a disk to a rectangle, it is likely that the size of the element must be increased as it switches from radius to full width.
- *Line*. Two dimensions must be given by the user: the length and the orientation of the line, where 0° is the horizontal. If your scratch is vertical, a line oriented by 90° might detect the scratch edges, but it is highly probable that it will be less efficient that the other shapes.
- *Diamond*. The diamond is a square tilted by 45°. The size of the structuring element corresponds to the length of its edges.
- *Square*. The size of the structuring element corresponds to the length of its edges.

Closing element

Once the cell sheet and its borders are reconstructed by the opening operation, the borders are cleaned by a closing operation. By using a structuring element smaller or of a different shape than the one of the opening operation, the closing operation allows to detail the cell sheet border. The default closing structuring elements was set to a disk having a diameter of 5 pixels. This allows to detail the shape of the border, which was already smooth if a disk was used for the opening operation too. If another shape, such as a rectangle, is used for the opening operation, it is likely that a small disk used for the closing will not allow to recover a smooth border. To obtain a neat result, the size of the disk closing element can be increased.

Of course, all the other structuring elements shapes available for the opening operation are also available for the closing operation. However, using them will likely lead to sharp edges unable to follow the smooth curves of the scratch borders.

Thresholding factor

After the computation of the gradient magnitude, a thresholding operation is applied to obtain a binarized image. Values above the threshold will be replaced by black pixels and values below by white pixels. This operation leads to the separation of the cell sheet elements in black from the background in white. A first threshold is determined

heuristically by the MATLAB algorithm to locate the edges of the cells. This threshold is then artificially lowered by multiplying it by a number between 0 and 1 as here, we want to detect not only the cells edges, but also their content, which corresponds to lower gradient magnitudes. This number is called the "thresholding factor". It is the main parameter influencing the scratch detection result. You can modify this parameter directly in the main app, and we suggest you start by modifying it before adjusting the other parameters. If the scratch appears wider than it should, try increasing the thresholding factor slightly. If the scratch appears smaller or some cells making the scratch border are not well detected, try decreasing the thresholding factor.

As in the main app, there are two ways to modify the thresholding factor in the *detailed changes* window: by using the slider or directly entering a precise value in the field at the left of the slider.

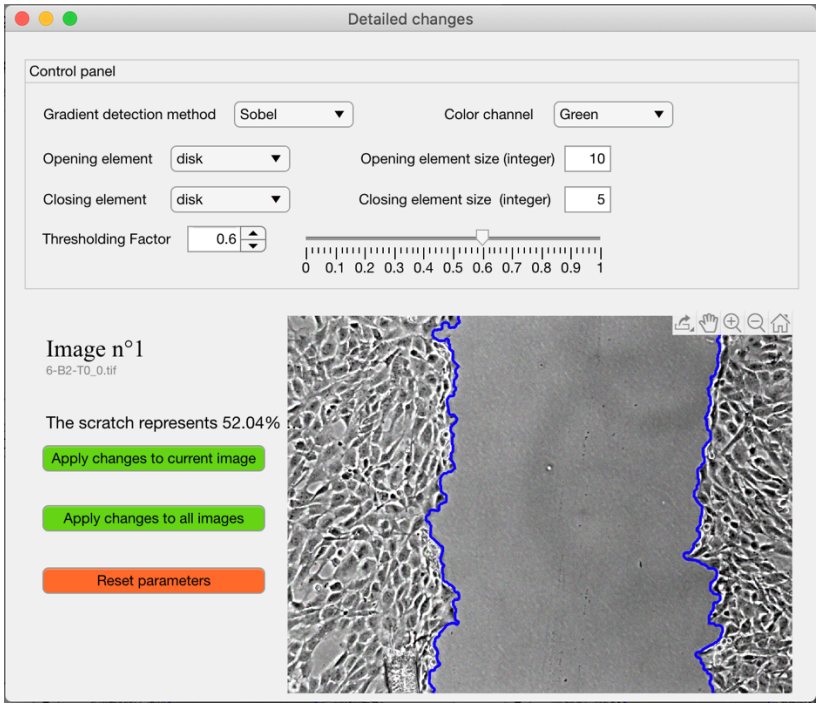

Figure 7. The "detailed changes" window allows to fine tune the scratch segmentation.

Output files

The *save results* button allows you to save all numerical results output by the algorithm: area of every analyzed scratches in % of total image or in mm^2 and, in case of a time serie: migration speed, coefficient of regression and the linear fit equation. Those can either be saved in a *.txt* file or in a *.xlsx* file, to allow an easy further processing of your data.

The ***save images*** button allows you to save all your initial images on which the contours of the scratch have been superimposed. All those images will be automatically saved at the location of your choice in a folder named ***result***.

The ***save graph*** button of the migration speed window allows you to save the graph showing the linear fit of your data.

Feel free to use any numerical result or images given by our application in your article by using the citation information given below.

Citation information

When conducting analysis by using SAMSCratch, please include the following citation in all related publications:

- Ahunon, L., Milano, F., Chevrier, A., & Lavertu, M. (2020). A novel image analysis algorithm reveals that media conditioned with chitosan and platelet-rich plasma biomaterial dose dependently increases fibroblast migration in a scratch assay. *Biomedical Physics & Engineering Express, 6*(6), 065021.

Contact

If you have any question or suggestion regarding the SAMScratch software (either source code or application), do not hesitate to contact us.

- Fiona Milano: Fiona.milano@polymtl.ca
- Laura Ahunon: Laura.ahunon&@polymtl.ca

We will be pleased to help you use the software, or to discuss with you about any improvement you can think about when using the software.