

UNIVERSIDADE ESTADUAL DE CAMPINAS
SISTEMA DE BIBLIOTECAS DA UNICAMP
REPOSITÓRIO DA PRODUÇÃO CIENTÍFICA E INTELECTUAL DA UNICAMP

Versão do arquivo anexado / Version of attached file:

Versão do Editor / Published Version

Mais informações no site da editora / Further information on publisher's website:

<https://www.sciencedirect.com/science/article/pii/S0169260719313914>

DOI: 10.1016/j.cmpb.2020.105476

Direitos autorais / Publisher's copyright statement:

©2020 by Elsevier. All rights reserved.

DIRETORIA DE TRATAMENTO DA INFORMAÇÃO

Cidade Universitária Zeferino Vaz Barão Geraldo

CEP 13083-970 – Campinas SP

Fone: (19) 3521-6493

<http://www.repositorio.unicamp.br>



PyScratch: An ease of use tool for analysis of scratch assays

Fernanda Garcia-Fossa^a, Vladimir Gaal^b, Marcelo Bispo de Jesus^{a,*}

^a Department of Biochemistry and Tissue Biology, Institute of Biology, University of Campinas, Campinas, São Paulo, Brazil

^b Applied Physics Department, University of Campinas, Campinas, São Paulo, Brazil



ARTICLE INFO

Article history:

Received 13 September 2019

Revised 28 February 2020

Accepted 20 March 2020

Keywords:

Scratch assay

Wound healing assay

Migration assay

Python

ABSTRACT

Background and objective: Image acquisition has greatly benefited from the automation of microscopes and has been followed by an increasing amount and complexity of data acquired. Here, we present the PyScratch, a new tool for processing spatial and temporal information from scratch assays. PyScratch is an open-source software implemented in Python that analyses the migration area in an automated fashion.

Methods: The software was developed in Python. Wound healing assays were used to validate its performance. The images were acquired using Cytation 5™ during 60 h. Data were analyzed using One-Way ANOVA.

Results: PyScratch performed a robust analysis of confluent cells, showing that high plating density affects cell migration. Additionally, PyScratch was approximately six times faster than a semi-automated analysis.

Conclusions: PyScratch offers a user-friendly interface allowing researches with little or no programming skills to perform quantitative analysis of *in vitro* scratch assays.

© 2020 Elsevier B.V. All rights reserved.

1. Introduction

Image analysis has become of paramount importance for molecular cell biology and medical research. While images have been processed for centuries, the popularization of automated microscopes has increased the amount and complexity of the data acquired. Therefore, analyzing and processing images requires specialized tools to accurately extract the data and interpret the results from high-throughput assays [1]. For example, cell migration is an assay vastly performed, because it is the underlying mechanism of many physiological and pathological cellular events. Measuring cell migration is relevant for understanding development and tissue modeling, angiogenesis, wound healing, and tumor development [2].

Although, in each field, the name used to describe this assay may vary (e.g., scratch assay, wound healing assay, migration assay), when performed in two dimensions they essentially measure how fast cells cover a pre-defined area. Usually, the experiment consists of creating a gap in the cell monolayer and tracking its closure over time [3]. To measure the velocity of cells migration, it

is necessary to acquire a substantial number of images; this step is a limiting factor to the analysis since it requires manual measurement. Along with the popularization of automated microscopes, data acquisition is no longer a limitation, allowing scientists to follow cellular events from fractions of a second up to days [4].

Several commercial and non-commercial tools are available to process the wound area. Two examples of non-commercial software are TScratch and MRI Wound Healing Tool [5,6]. TScratch measures the open space using edge-detection algorithm, which after a curvelet transformation, applies a threshold that separates the low-intensity open area and the high-intensity covered area [6]. This software requires MatLab Component Runtime (MCR) version 7.8 to run, which is commercial software, this may be a barrier to the use by non-programmer researchers. The second is MRI Wound Healing Tool installed as a macro on FIJI (an image processing package of ImageJ); it measures the homogeneity of the wound area, which is homogeneous compared to the cell-covered area [5–7]. However, these tools require full-time attention of the user to analyze and process the image data set. Including the time spending with the analysis, there is also the bias created by the researcher's belief when analyzing picture by picture.

Here we present PyScratch, a Python tool that quickly and autonomously analyses the area of migration assays. The Graphical User Interface (GUI) is friendly and easy to use, requiring no assistance, programming skills, nor specialized training; thus, enabling

* Corresponding author at. Department of Biochemistry and Tissue Biology, Institute of Biology, University of Campinas - UNICAMP, Rua Monteiro Lobato, 255, Campinas, São Paulo, 13083862, Brazil.

E-mail addresses: fefossa@gmail.com (F. Garcia-Fossa), vladimirgaal@hotmail.com (V. Gaal), dejesus@unicamp.br (M.B. de Jesus).

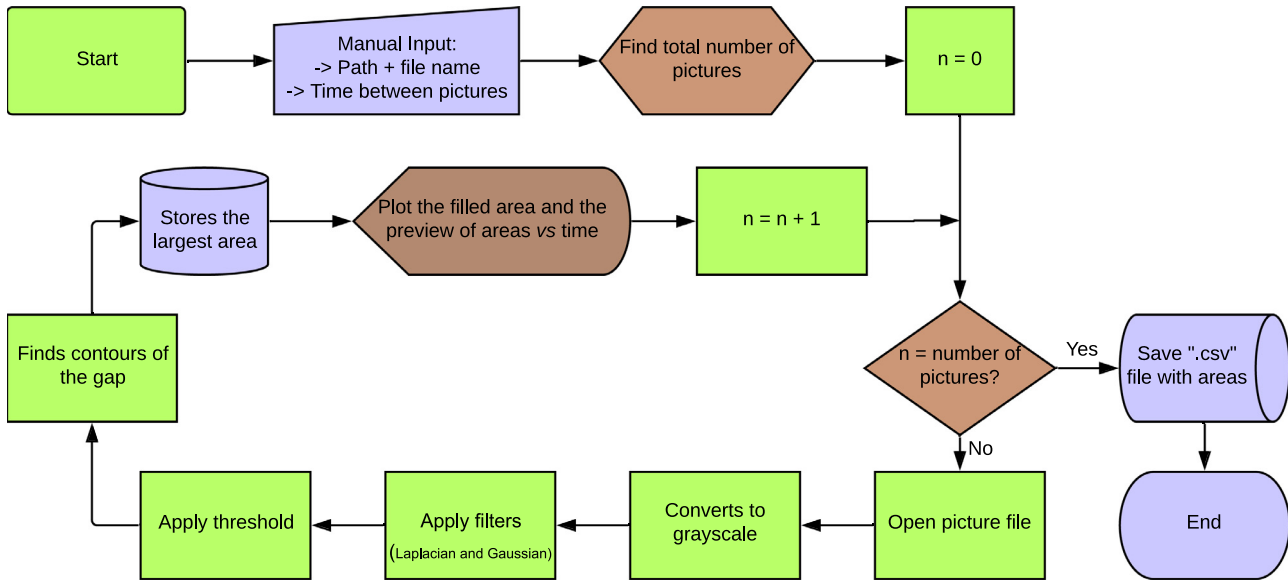


Fig. 1. Flow chart of PyScratch.

its use by non-programming researchers. The input files are a sequence of images acquired during the experiment, and the time interval between time points. The output is a comma-separated values (.csv) file, which stores tabular data in plain text, allowing the user to process the data using their usual routine. PyScratch is freely available and is an open source program that enables contributions from the scientific community.

2. Material and methods

2.1. Software architecture

We developed PyScratch as simple as possible to facilitate the researcher's daily practice. PyScratch was implemented in Python, using OpenCV 3 for image processing, which is an open source library that performs video and image analysis using C++ [8]. Additionally, we implemented a Graphical User Interface (GUI) to simplify the use by non-programmer users. Although GUI architecture is beyond the scope of this paper, we briefly described it in Fig. 2 for better clarification about the user interface functionalities.

PyScratch begins requesting the multiple sequential images, commonly used in time-lapse, which follow a pattern: your_file_name_###.tif, where ### represents the file sequence (000, 001, 002,...). With that information, the software identifies the number of images in that sequence. Thus, the images are loaded and transformed in grayscale to remove the RGB vectors (Fig. 1).

Next, three filters are applied: first, a Laplacian filter is used to enhance edges through zero-crossing the function. Based on the changes on pixel intensity close to the edge, we get the first derivative of the intensity obtaining a maximum; with the second derivative, we obtain the zero intensity that represents an edge [9]. So, the 2D Laplacian operator is defined by:

$$L(f) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

After applying the Laplacian filter, a Gaussian blur filter, which convolves the source image with kernels, is used to remove noise with high-frequency content, resulting in blurred edges. As parameters for Gaussian blur filter, are given the width and height of the kernel, and the standard deviation are computed by OpenCV itself

using the kernel size [8]. So, the 2D Gaussian blur equation is defined by:

$$G(x, y) = G(x) \cdot G(y)^t = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

The convolved images are now submitted to a threshold, which converts the picture into a binary image: the gap turns white (high intensity area), and the cells become black (low intensity area). Then, the OpenCV library can delimit the contour of the white area on the binary image, and PyScratch measures the largest area and stores the value in a vector. When the analysis is complete, the user stores the output areas using the Save button. Although the analysis happens in an automated fashion, the user can conveniently follow the measurements using the GUI interface. They can promptly access the images that are being processed, and the results being plotted on the fly, together with the progress bar to indicate the analysis development. At the end of the analysis, PyScratch saves a plot and a .csv database file, which contains three vectors: time in minutes, the area in pixels, and normalized area of each image and measurement (Fig. 1).

PyScratch provides all the pictures with the delimitation of the largest area as an output, which can be used to evaluate the analysis and used in publications to illustrate the measurements.

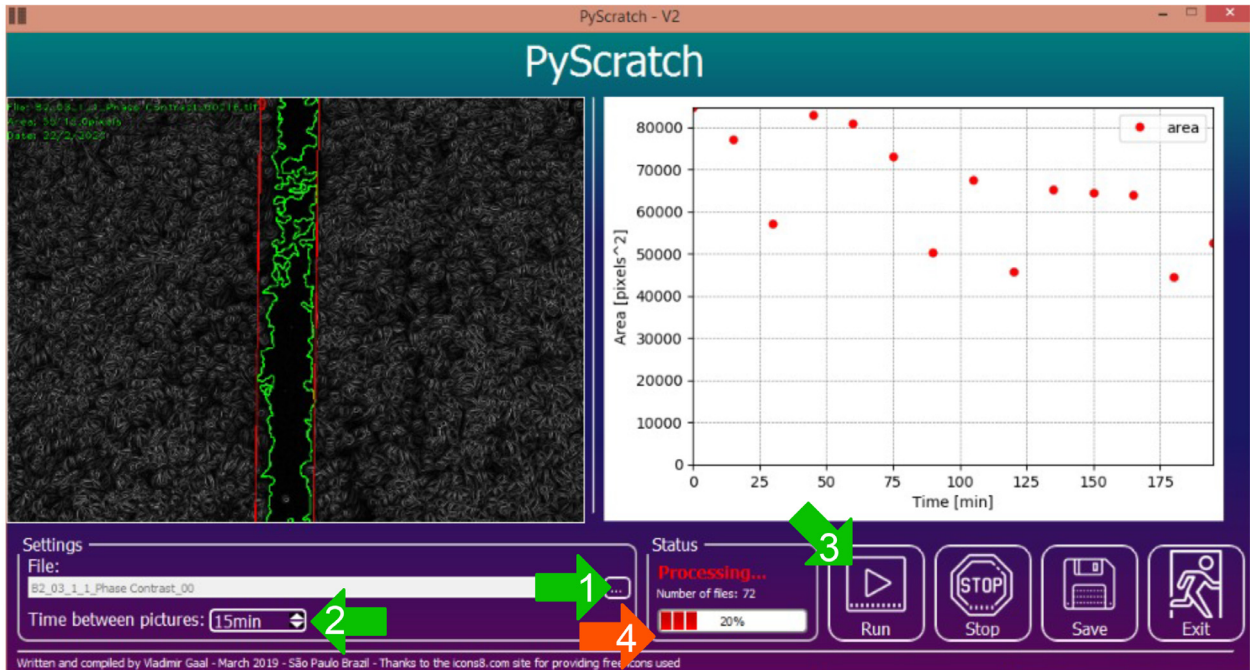
2.2. Software functionalities

The software recognizes the total number of files with the same pattern name, and processes all the images that belong to the same pattern. All filter parameters were optimized during the software development to analyze phase contrast images. We advise the user to acquire images using phase-contrast microscopy, since they often show a higher contrast between the cells and the gap than brightfield images. Therefore, facilitating the transformation of the pictures into binary images; thus, enabling the analysis. Other libraries than OpenCV are necessary to run PyScratch, for a complete list of the dependencies see Table 1.

2.3. In vitro experiments

Non-cancer cells (PNT1A) and cancer cells (PC-3) were seeded in density of 25,000; 50,000; and 100,000 cells per mL in a 12-well plate using RPMI 1640 (Gibco, Carlsbad, CA), 10% fetal bovine

A



B

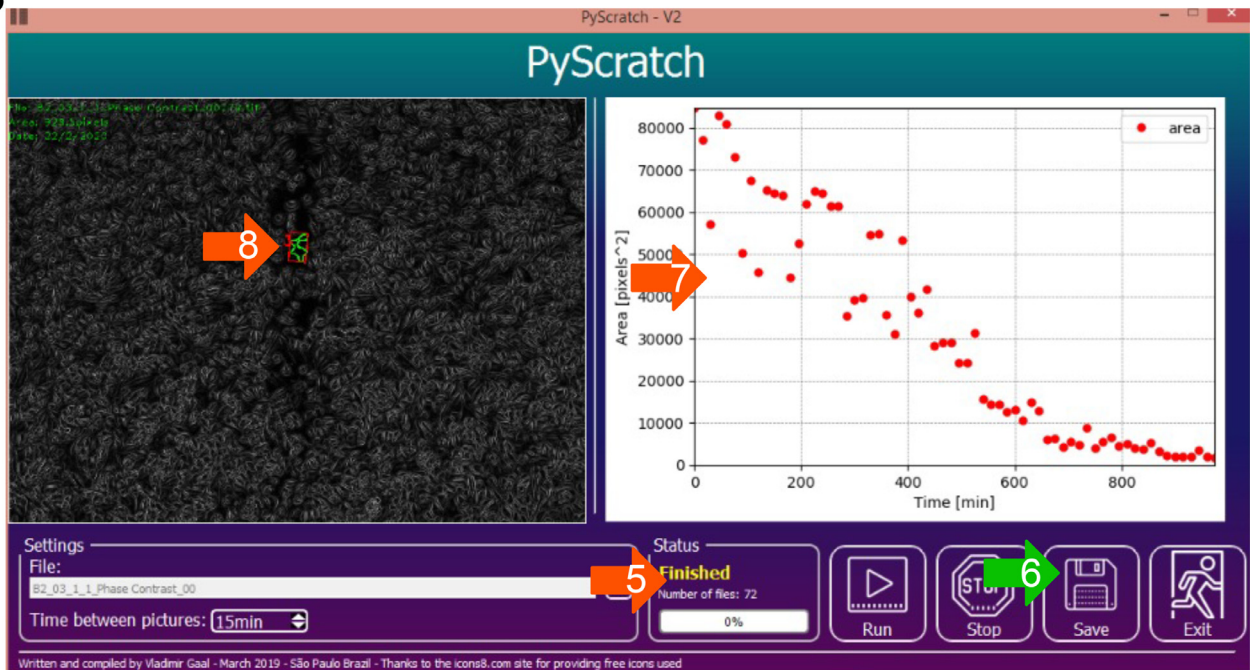


Fig. 2. GUI interface of PyScratch. Green arrows represent where the users' input (1, 2, 3, 6), while orange arrows indicate the software' outputs (4, 5, 7, 8). A. Software interface while running the analysis, where 1 = indicates where to select the first image to analyze, the program will load the sequential images; 2 = next, the user chooses the time between image acquisition; 3 = finally, the user should press 'Run' to initialize the analysis; 4 = user can follow the analysis through the number of files (sequential images) and the progress bar. B. Illustrates the GUI interface after the analysis is completed, where 5 = shows the 'Finished' status; 6 = where the user can save the output file, in .csv format; 7 = real-time graph that shows Area (pixels) versus Time (min.), and 8 = real-time measurement of the area in each sequential image, in red the ROI (region of interest) and in green the measured area.

serum (FBS) (Gibco, Carlsbad, CA) and 1% of penicillin/streptomycin (Gibco, Carlsbad, CA), and then incubated at 37 °C in a humidified CO₂ chamber. After 24 h, the medium was replaced with RPMI 1640 without FBS to synchronize cell's cycle. In the next day, the gap was created using a p200 pipette tip, and medium was replaced. The pictures were acquired with Cytation 5 (Biotek) every 15 min for 60 h. Over this period, 250 images were acquired

per treatment, resulting in 3000 images per experiment/plate. After PyScratch area analysis, the cell velocity was calculated with the Eq. (1) from [4]:

$$\text{Velocity} = \frac{|\text{slope}|}{2 \times \text{length}} \quad (1)$$

Table 1

Code metadata.

Code metadata description	
Current code version	V1 - Initial implementation V2 - GUI implementation
Permanent link to code/repository used of this code version	https://bitbucket.org/vladgaal/pyscratch_public.git
Legal Code License	MIT
Code versioning system used	Git (bitbucket)
Software code languages, tools, and services used	Python 3.6
Compilation requirements, operating environments & dependencies	Opencv_python - 3.4.7 Pandas - 0.24.2 PySide - 1.2.4 Matplotlib - 2.2.3 Numpy - 1.15.1
If available Link to developer documentation/manual	https://bitbucket.org/vladgaal/pyscratch_public.git
Support email for questions	vladimirgaal@hotmail.com

2.4. PyScratch performance

We also tested the PyScratch performance vs. macro MRI Wound Healing Tool from FIJI. Using an Intel processor Core i5 7200U, SSD 250GB, 8GB RAM computer, we analyzed eight batches of ten images (total of 80 pictures) from the scratch experiment using both PyScratch and FIJI, and we measured the analysis time for each using a timer (in seconds).

3. Results

To demonstrate PyScratch usability, we performed two experiments. First, we calculated the velocity rate of cells at different confluences (3.1). Second, we compared the time performance of PyScratch and the macro MRI Wound Healing Tool from FIJI using a 60 h experiment [5].

3.1. PyScratch in action

Many factors can influence the experimental outcome of migration assays; two of them are well known to play a crucial role in cell velocity: seeding density of cells and cell type [4]. Therefore, we decided to use these factors to evaluate the accuracy of PyScratch analysis. Thus, we measured the cell velocity at different seeding densities (25, 50, and 100 thousand cells per mL) of two cells lines, non-cancer cells (PNT1A, normal prostate cells) and cancer cells (PC-3, prostate cancer cells). Fig. 3A summarizes the experiment workflow, and Fig. 3B visually illustrates the difference in migration between cancer and non-cancer cells within 12 h. After processing this data with PyScratch, the results are shown in Fig. 3C. To calculate cell velocity, we first need the cell-covered area, which can be obtained by subtracting the total area by the gap area measured by PyScratch. Next, we can use a piecewise linear function to fit the equation using a Linear Trendline feature, and determine the slope of the regression line ($y = mx + b$, where m is the slope). The slope can be used to calculate the velocity of cell migration, which is the average velocity that cells move into the gap is equal to the slope over 2 times the length of the image, Eq. (1) [4]. PyScratch provides the area in pixels, allowing the user to use any microscope lens to acquire images and then convert to μm . To convert pixels to μm , we used the specification of the manufacturer lenses (in this case, 1 pixel corresponds to $0.620375 \mu\text{m}$). As the graph is plotted with the area in μm^2 , and time in minutes, we can assume that velocity is expressed in $\mu\text{m}/\text{minute}$.

Fig. 3C shows the different cell velocities calculated using PyScratch. We observe a positive linear correlation between seeding density and cell velocity in both cell lines, in accordance with [10] that showed the number of cells could influence the velocity rate. Additionally, the correlation was stronger in cancer cells, indicating that cancer cells migrate faster than non-cancer cells, an ex-

pected correlation [11]. This simple example shows how PyScratch can autonomously reach to accurate measurements.

3.2. PyScratch performance

To demonstrate the robustness and accuracy of PyScratch, we compared the analysis performed by PyScratch of the images acquired on the experiment described in the previous section with the semi-automated analysis using the macro MRI Wound Healing Tool (FIJI) (Fig. 4A). We detect no differences between the analysis performed by both software, indicating that PyScratch has the same accuracy as FIJI to the area measurements in different time-points (Fig. 4A). Also, we calculated the velocity of cell migration (Fig. 4B) using the areas provided by PyScratch and FIJI in Fig. 4A, obtaining similar values for velocity rates and standard deviation, showing that PyScratch autonomously can reach to same conclusions as FIJI.

We also decided to compare the time spent by the user in the analysis using both PyScratch and the macro MRI Wound Healing Tool (FIJI). This comparison took into consideration the different tasks performed by the user such as opening the folder that contains the pictures, loading the images into the program, analyzing the image and saving them to the desired folder. We used the same computer (Intel processor Core i5 7200U, SSD 250GB, 8GB RAM) to analyze eight batches of ten images (total of 80 pictures). PyScratch was approximately six times faster than FIJI to measure the area (Fig. 4C), since PyScratch only demands the intervention of the user to the input of images and to save the .csv file. These results show the benefit of automation in the time spent during analysis and likely to minimize human error.

4. Discussion

The power of image analysis determines the amount and the quality of information can be obtained from the experiment. Experiments investigating cell migration for different purposes are frequently performed; for example, a search for "Migration assay" resulted in 62,236 document results, for the same search using "Wound Healing Assay" in the Scopus database resulted in 19,198 document results and "Scratch assay" resulted in 4,611 documents results (05/04/2020). This experiment is often one of the first steps for studying cell migration of many researchers that investigates cancer, metabolism, protein expression, tumor invasion, angiogenesis, drug effects and many other subjects that can evolve from basic to applied research [12]. Hence, PyScratch presents as a software made to help non-programmer researchers focusing on reproducible, accurate, and in an autonomous fashion results.

Using PyScratch, we can eliminate, at least during the analysis, the bias created by the data collector's belief. In the scientific area, it is known that the researcher's expectations can create biases in

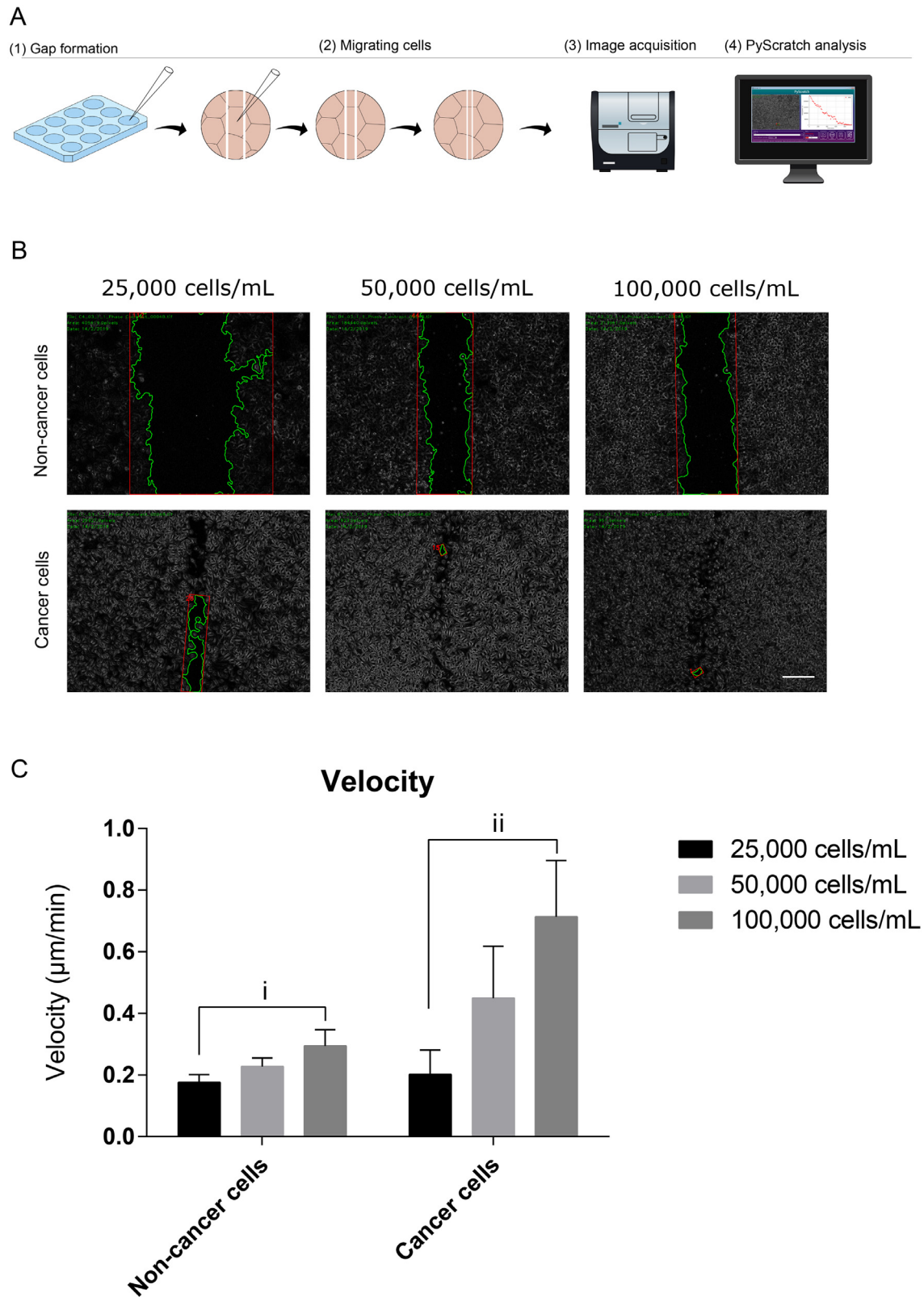


Fig. 3. Cell velocity at different seeding densities of cancer (PC-3) and non-cancer cells (PNT1A). After creating the gap, cells were incubated at 37 °C and images acquired every 15 min. during 60 h. A. Experiment design of a scratch assay. First, we create the gap between cells (1) using a p200 pipette tip. Then, the cells were incubated and migrate to the center of the gap (2). Images were acquired every 15 min. for 60 h (3) and analyzed using PyScratch (3). B. Representative images of non-cancer and cancer cells at 25, 50, and 100 thousand cells/mL at 12 h (720 min). Scale bar = 100 μm . Green lines represent the measured area and in red the region of interest (ROI). C. Velocity rate of non-cancer and cancer cells at 25, 50, and 100 thousand cells/mL. Each value represents the average \pm S.D. of three independent experiences ($n = 2$). i = Non-cancer cells at 25,000 \times 100,000 cells/mL; ii = Cancer cells at 25,000 and 100,000 cells/mL. $p < 0.05$, significative difference by one-way ANOVA, Tukey's test.

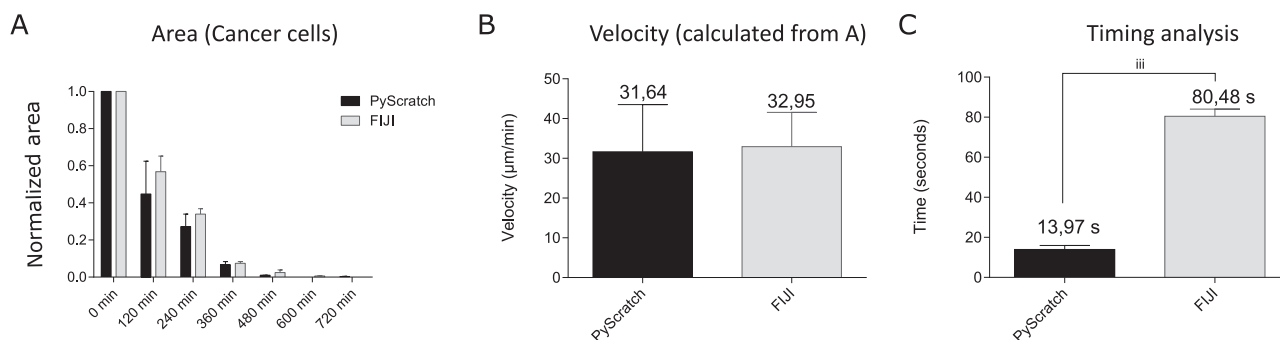


Fig. 4. Comparison between PyScratch and FIJI in scratch assay analysis. A. Normalized area of the gap at time 0, 120 min (2 h), 240 min (4 h), 360 min (6 h), 480 min (8 h), 600 min (10 h), and 720 min (12 h) of PC-3 cells at 100,000 cells/mL of confluence. Pictures were analyzed using both PyScratch and macro of FIJI MRI Wound Healing Tool [5]; B. Velocity rate of migrating cells calculated with the areas obtained in graph A. C. Comparison of the time that the user takes to: opens the file, starts the analysis and saves it. Each value represents the average \pm S.D. of three independent experiences ($n = 2$). iii = PyScratch x FIJI, $p < 0.05$ significant difference by 2way ANOVA, Sidak's test.

the moment of acquisition and analysis of data [13]. With lesser user intervention, the data quality is guaranteed as it reduces the experimenter effects in their research. The daily practice of the users is changed when, instead of being in the computer analyzing picture by picture, they can spend this time reading, evaluating results, and performing new experiments while PyScratch performs the analysis by itself.

5. Conclusions

In this paper, we have presented PyScratch, an open source tool for migration assay data processing with a user-friendly interface created to allow the use by researches with little or no programming skills. PyScratch is a crucial tool to facilitate the daily practice of the researcher and to exclude the bias created by the manual analysis increasing reproducibility and likely minimizing human error. Further development is expected since PyScratch is an ongoing project, and researchers can use it and contribute to the project. It is part of the author's interest to add new functionalities to the software with suggestions from the scientific community.

Declaration of Competing Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Acknowledgments

This research was supported by Sao Paulo Research Foundation (FAPESP) (grant #2014/03002-7), and Multi-User Equipment Program (grant #15/06134-4). This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. We thank Dr. J. M. Nascimento and R. Brenzikofer for their careful reading and feedback on the paper.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at [doi:10.1016/j.cmpb.2020.105476](https://doi.org/10.1016/j.cmpb.2020.105476).

References

- [1] Z. Yin, H. Lan, G. Tan, M. Lu, A.V. Vasilakos, W. Liu, Computing platforms for big biological data analytics: perspectives and challenges, *Comput. Struct. Biotechnol. J.* 15 (2017) 403–411, doi:[10.1016/j.csbj.2017.07.004](https://doi.org/10.1016/j.csbj.2017.07.004).
- [2] S.A. Eming, P. Martin, M. Tomic-Canic, Wound repair and regeneration: mechanisms, signaling, and translation, *Sci. Transl. Med.* 6 (2014) 265sr6–265sr6, doi:[10.1126/scitranslmed.3009337](https://doi.org/10.1126/scitranslmed.3009337).
- [3] A. Stamm, K. Reimers, S. Strauß, P. Vogt, T. Scheper, I. Pepelanova, In vitro wound healing assays – state of the art, *BioNanoMaterials* 17 (2016), doi:[10.1515/bnm-2016-0002](https://doi.org/10.1515/bnm-2016-0002).
- [4] J.E.N. Jonkman, J.A. Cathcart, F. Xu, M.E. Bartolini, J.E. Amon, K.M. Stevens, et al., An introduction to the wound healing assay using live-cell microscopy, *Cell Adhes. Migr.* 8 (2014) 440–451, doi:[10.4161/cam.36224](https://doi.org/10.4161/cam.36224).
- [5] C.A. Schneider, W.S. Rasband, K.W. Eliceiri, NIH Image to ImageJ: 25 years of image analysis, *Nat. Methods* 9 (7) (2012) 671–675, doi:[10.1038/nmeth.2089](https://doi.org/10.1038/nmeth.2089).
- [6] T. Gebäck, M.M.P. Schulz, P. Koumoutsakos, M. Detmar, TScratch: a novel and simple software tool for automated analysis of monolayer wound healing assays: short technical reports, *BioTechniques* 46 (2009) 265–274, doi:[10.1214/000113083](https://doi.org/10.1214/000113083).
- [7] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, et al., Fiji: an open-source platform for biological-image analysis, *Nat. Methods* 9 (2012) 676–682, doi:[10.1038/nmeth.2019](https://doi.org/10.1038/nmeth.2019).
- [8] Culjak I., Abram D., Pribanic T., Dzapo H., Cifrek M. A brief introduction to OpenCV. 2012 Proc. 35th Int. Conv. MIPRO, 2012, p. 1725–30.
- [9] G. Bradski, A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, O'Reilly Media, Inc., 2008.
- [10] N. Molinie, A. Gautreau, Directional collective migration in wound healing assays, in: Gautreau A., editor. *Cell Migr. Methods Protoc.*, Springer, New York, NY, 2018, pp. 11–19, doi:[10.1007/978-1-4939-7701-7_2](https://doi.org/10.1007/978-1-4939-7701-7_2).
- [11] F. Linke, S. Zaunig, M.M. Nietert, F. von Bonin, S. Lutz, C. Dullin, et al., WNT5A: a motility-promoting factor in Hodgkin lymphoma, *Oncogene* 36 (2017) 13–23, doi:[10.1038/onc.2016.183](https://doi.org/10.1038/onc.2016.183).
- [12] A. Grada, M. Otero-Vinas, F. Prieto-Castrillo, Z. Obagi, V. Falanga, Research techniques made simple: analysis of collective cell migration using the wound healing assay, *J. Invest. Dermatol.* 137 (2017) e11–e16, doi:[10.1016/j.jid.2016.11.020](https://doi.org/10.1016/j.jid.2016.11.020).
- [13] L. Holman, M.L. Head, R. Lanfear, M.D. Jennions, Evidence of experimental bias in the life sciences: why we need blind data recording, *PLoS Biol.* 13 (2015), doi:[10.1371/journal.pbio.1002190](https://doi.org/10.1371/journal.pbio.1002190).