ACS OMEGA

http://pubs.acs.org/journal/acsodf

Article

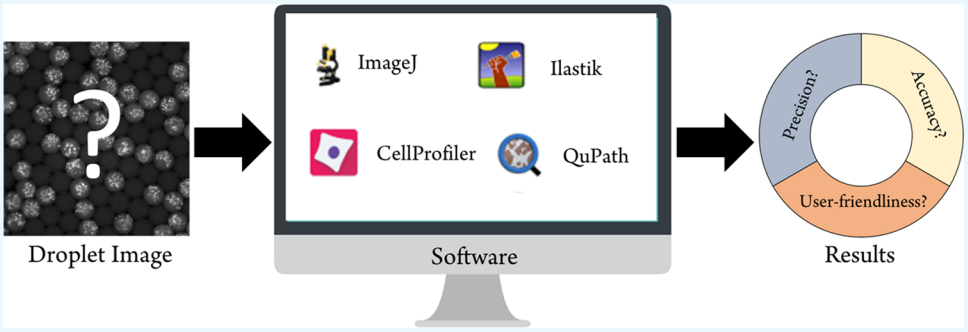
Investigation of Different Free Image Analysis Software for High-Throughput Droplet Detection

Immanuel Sanka, Simona Bartkova, Pille Pata, Olli-Pekka Smolander, and Ott Scheler*

Cite This: *ACS Omega* 2021, 6, 22625−22634

Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information



ABSTRACT: Droplet microfluidics has revealed innovative strategies in biology and chemistry. This advancement has delivered novel quantification methods, such as droplet digital polymerase chain reaction (ddPCR) and an antibiotic heteroresistance analysis tool. For droplet analysis, researchers often use image-based detection techniques. Unfortunately, the analysis of images may require specific tools or programming skills to produce the expected results. In order to address the issue, we explore the potential use of standalone freely available software to perform image-based droplet detection. We select the four most popular software and classify them into rule-based and machine learning-based types after assessing the software's modules. We test and evaluate the software's (i) ability to detect droplets, (ii) accuracy and precision, and (iii) overall components and supporting material. In our experimental setting, we find that the rule-based type of software is better suited for image-based droplet detection. The rule-based type of software also has a simpler workflow or pipeline, especially aimed for non-experienced users. In our case, CellProfiler (CP) offers the most user-friendly experience for both single image and batch processing analyses.

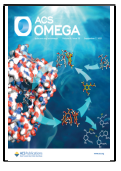
INTRODUCTION

Droplet microfluidics has become a powerful tool for high-throughput analysis over the last few decades.[1] It allows compartmentalization of samples in massive parallelization.[2] This high-throughput technique is also compatible with different analytical technologies, e.g., mass spectrometry.[3] Droplets are often applied for high sensitivity nucleic acid diagnostics[4] or different microbiological studies.[5] For instance, the tool has also been used to perform high-throughput screening for protein crystals,[6] DNA quantification by digital droplet polymerase chain reaction (ddPCR),[7,8] detecting viable bacteria and heteroresistance in antimicrobial experiments,[9,10] or performing experiments with mammalian cells.[11]

Image-based analysis has often been used in droplet microfluidic experiments.[12] The analysis has been implemented in different types of image data, from single static image up to real-time data, either by bright-field or fluorescence microscopy.[13] This approach has been used for a wide range of experiments, such as bacterial surveillance of foodborne contamination,[14] screening of specific substrates,[15] single-cell analysis,[16] and detecting viable bacteria or viruses (e.g., SARS-CoV-2).[17,18] Image-based droplet analysis (IDA) often requires specific skills in programming that are not widely available in non-specialist laboratories. Most of the published articles in droplet detection use scripted programs, such as Circular Hough Transform in Python programming language,[19] Mathematica,[20,21] Scikit-image in Python,[22] Image Processing Toolbox from MATLAB,[23] OpenCV and Keras in Python,[24] and OpenCV in C++.[25] There are some user-friendly software that may be used for droplet microfluidic image analysis, such as the Zen imaging program[26] and NIS-Elements from NIKON.[14] However, these kinds of programs are only commercially available.

There is a need for widely accessible and user-friendly IDA tools for image-based droplet analysis. Open-source software is available and can be used to detect and/or analyze droplets. For example, ImageJ software has been used to analyze image

Received: May 21, 2021
Accepted: August 13, 2021
Published: August 26, 2021



https://doi.org/10.1021/acsomega.1c02664
ACS Omega 2021, 6, 22625−22634

data in general[27] including droplets,[28] or CellProfiler (CP), which was developed to identify and measure various bioimage data.[29] Even though some published articles mention the use of the software, information regarding their workflow is limited (the data is often missing from publications). This would confuse early-stage researchers with little or no experience in image analysis, specifically for image-based droplet detection using no programming skills. However, novel workflows can be constructed by combining functions, modules, or pipelines from different software, like building a puzzle.[30]

Here, we (i) demonstrate how to use different software for the analysis of droplet images in static 2D images and (ii) explore the differences and similarities of workflows in the different software from the perspective of detecting, counting, and measuring the properties (including but not limited to droplet number, diameter, fluorescence intensity of droplets, etc.) using four selected software (Table 1).

Table 1. General Characteristics of Selected Software[c]

Requirement	CellProfiler	ImageJ	Ilastik	QuPath
Version	4.0.3	1.52p	1.3.3	0.2.3
Operating system	Win, Mac	Win, Mac	Win, Mac	Win, Mac
Bit machine	64 and 32	64 and 32	64	64
RAM and hard disk space	4 Gb & NA	NA & NA	8 Gb & NA	4 Gb & NA
Written in	Python	Java	Python	Java
Compatible file format	wide[a]	wide[a]	wide[a]	wide[a]
Output[b]	v	v	v	v
Available plugins	v	v	-	-
Documentation	v	v	v	v
Batch processing	v	v	v	v

[a]Wide is the general image file type, such as TIFF, JPEG, PNG, etc. [b]Generates object size, pixel intensity, circularity, object position, etc. [c]v = available, − = not available

RESULTS AND DISCUSSION

Software Selection and Workflow Construction. The most popular software for image analysis are ImageJ (IJ), CellProfiler (CP), Ilastik (Ila), and QuPath (QP). Here, we use Twitter and Scopus repositories to find the popularity of the software in the field of image analysis. Twitter has been used for research purposes before.[32] We found that social media also give researchers the opportunity to "push" their findings and correlate them to a greater citation.[33] To find the popularity, we executed Twint[34] Python script using each of the software's name as the keyword. For finding the results from Scopus' repository, we also used the same keyword. Both searches were performed to acquire data from January 1, 2010 to December 31, 2020. Based on the Scopus and Twitter search (obtained on February 11, 2021), we showed the sum of "tweets" or 160-character max of text from Twitter and the sum of Scopus search in scatter plot (Figure 1A). The most popular software are ImageJ,[27] CP,[35] Ilastik, and QuPath, in blue, red, cyan, and green color, respectively. Ilastik uses the concept of supervised machine learning in their workflow,[36] and QuPath has been used as a whole slide image analysis tool.[37] We continued with these four popular software tools and used them to detect droplets on the image dataset previously described by Bartkova et al.[31] (Figure 1B). Then, we took a deeper look into their workflow and assessed their performance with different key parameters (Figure 1C).

Rule-Based and Machine Learning-Based Software for Droplet Detection. We divided the selected software into two groups (rule-based and machine learning-based) according to their workflow. In the rule-based software group (CP and ImageJ), users have to manually provide settings for the program to select the pixels of interest with numeric or known parameter in order to detect droplets. In the machine learning-based group (Ilastik and QuPath), on the other hand, users may select the areas of the image (labeling) and manually annotate them as objects of interest (e.g., droplets or background) for pixel classification. Based on these characteristics, we described the abstraction of the process with three increasing levels and used it to direct the image-based droplet detection.

Pre-processing, Processing, and Post-processing Concepts. We used the terms (i) pre-processing, (ii) processing, and (iii) post-processing. (i) In pre-processing, we modified, adjusted, and prepared the image data for further use. For instance, we performed pre-processing to duplicate the image data, introduce features, and make annotation(s) on the image. In addition, we also include the image setup, such as image upload, metadata setting, and supporting option before processing the image data. For instance, we also included the macro record in IJ and the metadata setup in CP.

(ii) In processing, we conducted segmentation or pixel partitioning based on color, intensity, or texture along with droplet detection or counting process.[38] Usually, processing steps may help users obtain a specific type of data.[39] In our case, we introduced thresholdring to distinguish between the background (dark) and the foreground (droplets). For the details, CP came in handy and only needed one module named "IdentifyPrimaryObject", which contained some options to detect droplets. This included thresholding, smoothing, segmentation, and automatic selection. In ImageJ, processing steps had three options: "Thresholding", "Watershed", and "Analyze Particle". Similar to CP, these three steps will provide selections to detect the droplets. In the processing part, Ilastik had to process "Thresholding", "Object Feature Selection", and "Object Classification" for selecting the droplets and discarding the background. In QuPath, we found all of these features in "Pixel Classifier". The settings included a classifier from an artificial neural network with multilayer perception (ANN_MLP)[40] with high resolution, using four multiscale features (Gaussian, gradient magnitude, Hessian determinant, and Hessian max eigenvalue) with probability as an output.

(iii) For the last step, in post-processing, we prepared data extraction or generation for further use, for example, to generate a table of data or type of images for visualization. In CP, this last step was performed with "OverlayOutlines", "OverlayObject", "DisplayDataOnImage", and "ExportToSpreadsheet". These modules generated the images and results in CSV format. The order was similar in ImageJ and Ilastik, but the option was available in "ROI Manager" and "Export", respectively. In QuPath, the results can be obtained by exporting annotations from detected objects or called as labeled images. We used the Groovy script to generate this result using commands in "Workflow" tab. Groovy is a compiled language that can be integrated seamlessly with Java. However, it has some semantic and practical differences, especially regarding syntax.[41] For a brief workflow/pipeline, we provide the scheme of third level complexity in Figure 2.

https://doi.org/10.1021/acsomega.1c02664
ACS Omega 2021, 6, 22625−22634

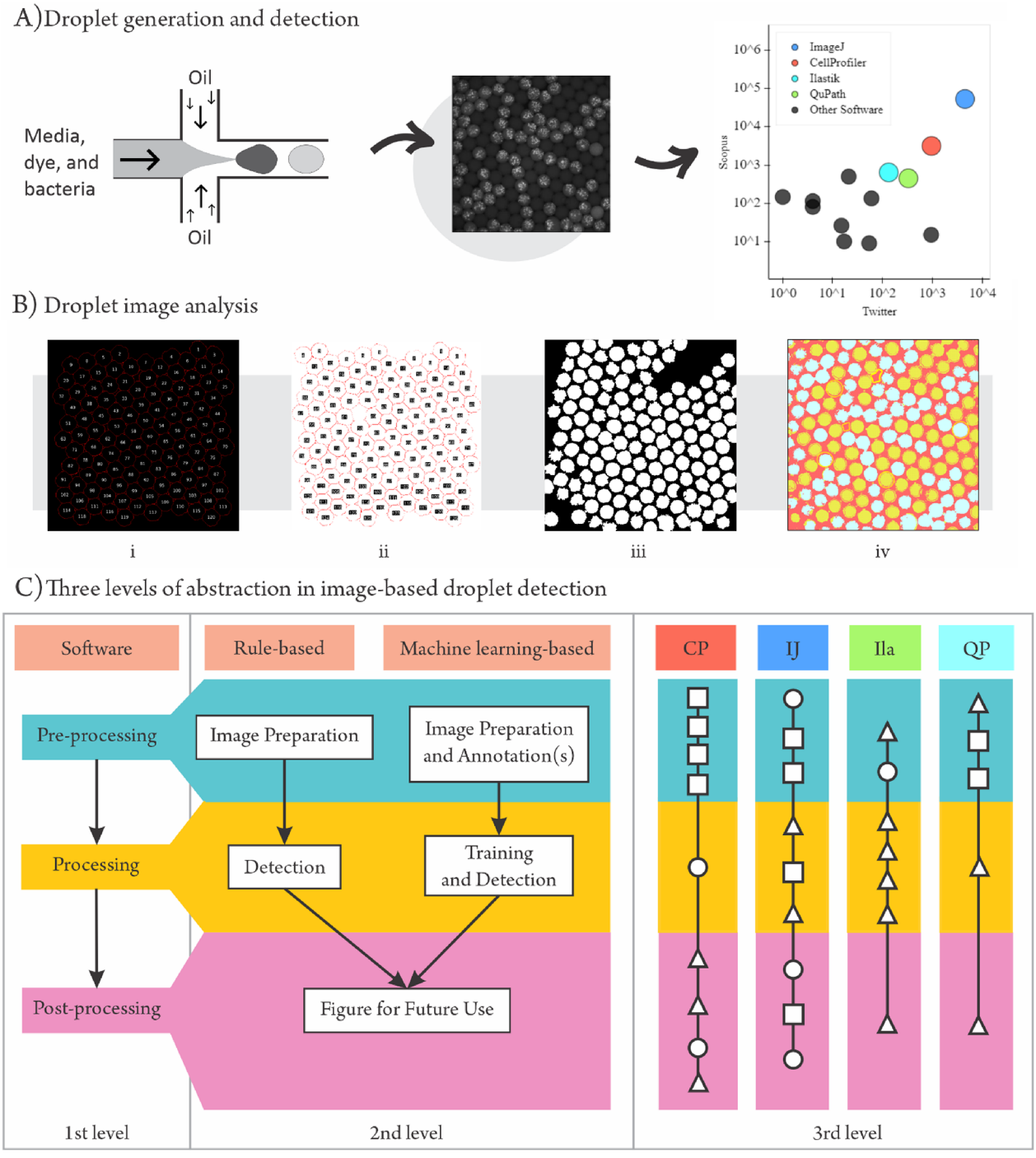


Figure 1. Schematic of droplet generation and image analysis of a single image. (A) We generated water-in-oil droplets using a flow-focusing microfluidic chip (left). We used a fluorescence microscope to obtain "raw images" of droplets that contained fluorescence producing bacteria (middle). For the analysis of droplet images, we used the four most popular image analysis software that were selected according to hits in social media (Twitter) and Scopus search (obtained on February 11, 2021) (right). (B) Droplet detection comparison among (i) ImageJ (IJ), (ii) CellProfiler (CP), (iii) Ilastik (Ila), and (iv) QuPath (QP). (C) We divided the image processing software into two groups (rule-based and machine learning-based) and explored their logic and working principle on three levels of abstraction. (1) The first level shows that used software are very similar in their basic image processing logic. They usually have three processing stages in their image analysis logic: pre-processing, processing, and post-processing. (2) The second level shows distinction between two groups of software in droplet detection: rule-based, where users define how to detect droplets by giving specific parameters (e.g., threshold or size), or machine learning-based, where users classify/annotate grouping of pixels on an image. (3) The third level shows a number of different steps and modules in processing stages. For the object on the left side of each workflow, we use a triangle to determine the module with only one option, rectangle for the module with two to eight options, and circle for the module with more than eight options.

CP Has the Highest Accuracy and Precision. By comparing the results with manually counted droplets (7145), we investigated the ability of the analyzed software to detect droplets. We only counted the droplets that did not

https://doi.org/10.1021/acsomega.1c02664
ACS Omega 2021, 6, 22625−22634

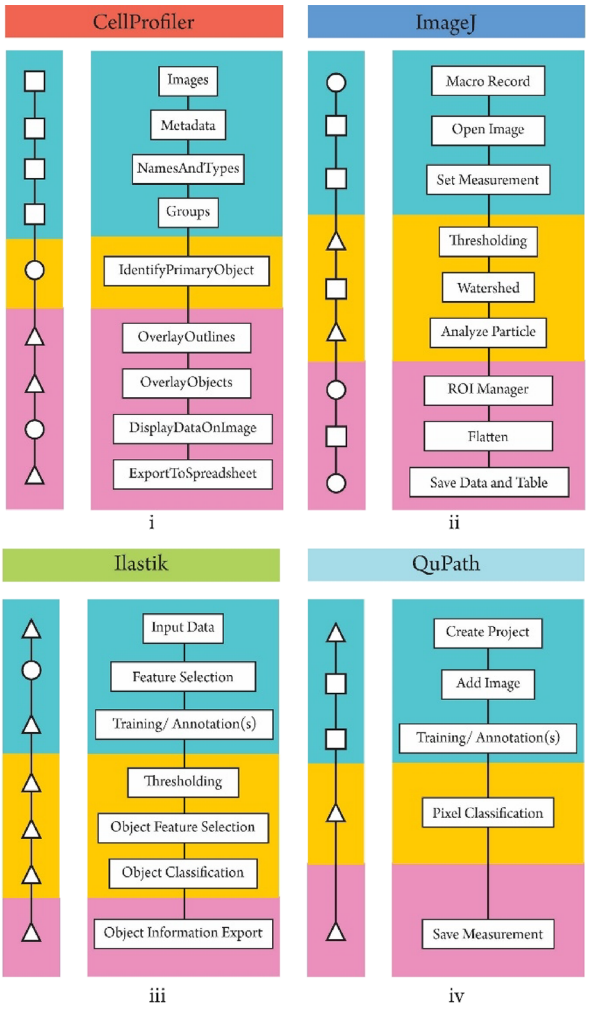


Figure 2. Detailed third level of abstraction for image-based droplet detection using (i) CellProfiler, (ii) ImageJ, (iii) Ilastik, and (iv) QuPath. The symbols represent how many options are within each module, referring to the previous figure where the triangle, rectangle, and circle represent one, two to eight, and more than eight options, respectively. The background colors correspond to pre-processing (cyan), processing (yellow), and post-processing (magenta).

touch the image border and did not make a bundle (joint droplets because of failed segmentation). We performed sensitivity and specificity tests using True Positive (TP), False Positive (FP), and False Negative (FN) values based on the comparison with manual counting.[42] The TP confirms the positive droplet detection in the data. For FP, the value is obtained by finding false droplet detection or underestimation (type I error). In FN, the software does not detect the droplet or performs overestimation (type II error). We defined TN as the background (black = 0). After the calculation, we obtained the accuracy ((TP + TN)/(TP + TN + FP + FN)) and precision (TP/(TN + TP)) from the detection. This accuracy explains the ratio between the correct droplet detection and total number of droplet detection. On the other hand, precision describes the probability to produce the correct droplet detection in total positive detection.[43−45] The accuracy of each detection ranges from 74.7 to 96.2%. One of the software managed to generate a precision of up to 99.8% (Table 2).

Low-Image Quality Gives More False Detection. From Figure 3, we can see how each group shares similar errors in

Table 2. CP Gives the Highest Accuracy and Precision

Category	CellProfiler	ImageJ	Ilastik	QuPath
% Accuracy	96.2%	92.7%	74.7%	80.9%
% Precision	99.8%	96.3%	80.2%	83.1%

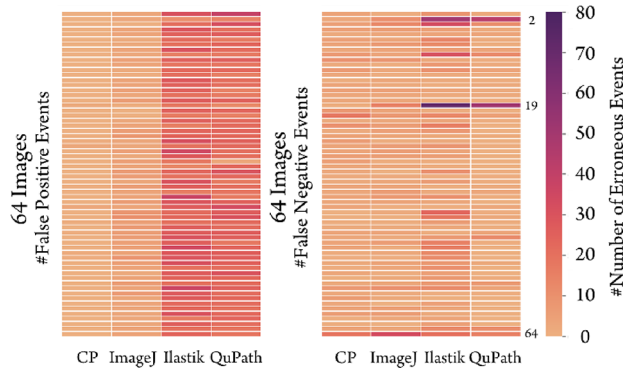


Figure 3. Droplet detection errors are higher in machine learning-based software in the diagnostic test. The figure shows the False Positive (FP, wrong detected droplet) and False Negative (FN, wrong undetected droplet) events per image. Each block represents event or image error detection. The scale shows the number of errors (dark = high and bright = low).

every event (detection per image). We compared the false detection results (both FP and FN) from each of the software. We found that the rule-based group (CP and ImageJ) have less false detection compared to the machine learning-based group (Ilastik and QuPath). However, Ilastik and QuPath received high error because they do not have filters to eliminate the droplets that touch the border, and some droplets are falsely detected as joint droplets (Figure S1). Figure 3 also shows images, which may have bad quality for droplet detection. For instance, image numbers 2, 19, and 64 depict the highest error values from all four software. Notwithstanding, CP outperforms the other software and has both high accuracy and precision.

Each Software Requires Different Workflows for Batch Processing. CP is the most suitable software for batch analysis or high throughput analysis. In CP, we can analyze a whole set of images with a press of a single button "Analyze Images" on the main menu. The software will process available images uploaded in the "Images" module (default module). We tested and used the batch processing option to analyze 64 images straight after we had our pipeline/workflow set. In ImageJ, we processed the batch analysis using a recorded macro by single image analysis. We also performed some macro script cleaning (e.g., closing unnecessary tabs during the process), which was written in the macro recorder. After cleaning, we selected the input and output folders and performed batch processing through the "Process" tab. For Ilastik, we executed batch processing after the last option of the pipeline. We just needed to upload the images and started the "Process all files". QuPath demanded macroprogramming commands for executing batch analysis. However, this software provided an automated script generator that simplified the macro record to perform batch analysis. ImageJ and QuPath required a macro script for batch analysis. Even though this macro script was easy to do, creating a macro script for the first time could become an obstacle for researchers who are not familiar with any programming language or practices.[46] From

https://doi.org/10.1021/acsomega.1c02664
ACS Omega 2021, 6, 22625−22634

our viewpoint, CP and Ilastik had the most user-friendly interface for batch processing because they provide the option to scale up after single image pipeline construction and do not require any programming steps. Therefore, finding any additional button or tab to batch process the images was unnecessary. On the other hand, process and scripting were required in ImageJ and QuPath.

Modularity Gives More Flexibility in Developing Pipelines. Rule-based class software are flexible and have modular options in processing image(s). As rule-based tools, CP and ImageJ offered options that could be added and removed depending on the user's preferences, such as the type of thresholding algorithm, filters, and other modules. In machine learning-based software, the features were embedded in the pipeline and had limited availability for additional settings. For example, Ilastik had some pre-defined pipelines: one of them was object classification and pixel classification.[36] These two were fixed in the interface of Ilastik and may be rearranged only through Python programming. From Figure 1C, the third level of complexity also represents the modularity in which Ilastik and QuPath were more limited than CP and ImageJ. For instance, CP had the "IdentifyPrimaryObject" module that could be duplicated in one pipeline, while in Ilastik, "Thresholding" could be performed only once within the pre-defined workflow. This complication placed Ilastik as the least flexible tool followed by QuPath.

Batch Processing Time Is Shorter in Java-Based Software. Macro programming language affects the software processing time, particularly in batch analysis. CP or ImageJ expected less computational power for the use since they did not implement machine learning classification methods in our pipelines. The use of machine learning requires training and features implementation that requires more computational power.[47] The rule-based software used object logic classification[48] and did not require training set to test the defined parameters, e.g., size of the object or maximum length of the object. In QuPath and Ilastik, the classification depended on a supervised machine learning process.[36,48] We used manual annotations (droplets and background) in making the classifier before processing whole pixels. We also previously compared the minimum hardware requirements for each of the tools (Table 1). Based on the comparison, ImageJ was the only one that did not put any minimum requirement on the random-access memory (RAM). We also expected that the machine learning-based software might take more time to process the whole set of images. Therefore, we also tried running the whole pipeline and comparing the performance from each of the tools. We tested each pipeline with the same computer having an Intel Core i3-9100F processor, 8GB RAM, NVDIA GeForce GTX 1660 SUPER, 120Gb SSD PANTHER and running in a Windows operating system. In our setting (with the same environment and background setting), we found that QuPath and ImageJ perform faster than CP and Ilastik in batch processing (Figure 4). The experiment was conducted by running the same pipeline 10 times to find the deviation as well. Tool's batch processing language (macros) may cause this difference. At the beginning, we expected Ilastik and QuPath to have longer processing time than CP and ImageJ because of the machine learning-based processing. However, ImageJ and QuPath performed faster than others. In principle, there are two types of program that bioinformaticians use: compiled and interpreted.[49] ImageJ and QuPath use Java based (macros) code that is compiled once before the program processes the

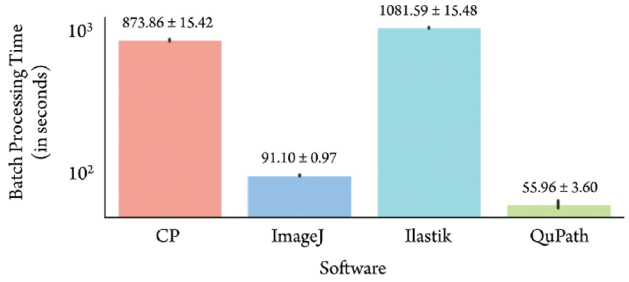


Figure 4. Java-based software has a shorter processing time in the droplet detection scenario. We ran the droplet detection pipelines in the same computer with the same dataset. Error bars show standard deviation between 10 replicate analysis runs with the same set of 64 images.

batch analysis. Presumably, this allows the program to run faster. On the other hand, CP and Ilastik use Python to process batch analysis. In Python, variables and functions will be run through an interpreter every time the program needs to process the task, in our case, to detect droplets in every image. Regardless, we do not have enough evidence to claim that the type of software may shorten the processing time. Nonetheless, a speed comparison of different types of language (including Python and Java) to run the same command showed that implementation in Java performs up to 20 times faster than in Python.[50] We also note that different hardware can alter the performance of software in different settings, but the relative ratios of needed computing resources should be similar.

Documentation Is Important in Pipeline Development. CP and ImageJ have sufficient examples and documentation for novice users. Each of the software provides documentation and examples for guiding their users. CP and ImageJ have been developed since 2005 and 1987, respectively.[46,51] Therefore, these rule-based software have more users and examples, e.g., ImageJ has a distribution for compiling the biological image analysis plugins called Fiji.[27] CP also provides some tutorials, examples, and other documentation on their website, e.g., detecting different cell morphology and tracking objects (www.cellprofiler.org). On the contrary, Ilastik and QuPath have limited documentation for accompanying new users. However, these two software also have extensive documentation, including their manuals and tutorials for both novice and advanced users at their website (https://ilastik.org/documentation and https://qupath.readthedocs.io). Additionally, there are some forums such as image.sc forum (forum.image.sc) that are actively helping other bioimage researchers or software users.

Plugins May Ease Users to Perform Specific Image-Based Detection. Plugins in CP and ImageJ can be used as an extensible option in processing images. Plugins or add-on can be used to improve default options within the software. These may be utilized by other software developers. As an additional option, plugins may help the user implement specific cases of detection. Before Ilastik and QuPath were developed, ImageJ had plugins called Trainable WEKA Segmentation that, in principle, works similarly to machine learning-based software.[52] In CP, plugins are also available. For instance, we found one plugin that analyzes mass cytometry (multiplexed images) called ImcPluginsCP.[53] Here, we did not add any plugins to detect droplets and we used similar settings to see the tool's ability to detect and count droplets. The extension software for CP, CellProfiler Analyst (CPA),[54] could

https://doi.org/10.1021/acsomega.1c02664
ACS Omega 2021, 6, 22625−22634

be an option to enhance droplet detection, which has been described briefly in our previous research.[31] Based on our classification, CPA belongs to machine learning-based software because users need to supervise or train the data at the beginning. However, this software is not a standalone software and requires feature extraction or properties file that contain the observed data from CP.[31,55]

Image Components Take Important Role in Processing. Rule-based software are more suitable for analyzing droplet microfluidic image data. Rule-based software provide more options, e.g., to disregard the object that touches the border/frame, which resulted in high accuracy and precision. On the other hand, the machine learning-based software required more optimization to train the classifier. We only used 12 lines (5 lines for determining droplets and 7 lines to define borders between droplets and background) to supervise each class (background and droplet). Each line represents the pixels for each group. This pixel manual selection works better if the image has similar properties in majority and represents the pixel distribution of an object, for example, borders between droplets and empty droplets. Even though the droplet's border looks the same across the image, the pixel distributions are varied. We picked more lines to define borders. We also needed extra time to train the classifier (in minutes) when setting the machine learning-based software to determine the 12 lines. However, this cannot represent all of the properties and may result in joint droplets. To overcome this, a larger training set and improvement of the classifier would presumably give a better result. As an image processing tool, the machine learning-based software like QuPath has a more specific purpose. Moreover, this software was created to accommodate whole slide image and large image data analyses, specifically for complex tissue images.[37] However, a comparison has been made between QuPath and CP coupled with CPA.[48] The comparison also shows the pros and cons between the rule-based and machine learning-based software in renal tissues. Furthermore, ImageJ, CP, Ilastik, and QuPath have shown their capability in detecting droplets and generating the results as standalone tools.

Data Acquisition Can Be Embedded in the Pipeline. Droplet detection is often used as a preliminary step in droplet microfluidic experiment. It is possible to expand the pipeline for further analysis, e.g., bacteria detection,[31] enzyme reaction measurement,[56] chemical purification analysis,[57] and metal extraction.[58] This step is usually performed to extract the different aspects of a droplet (size, texture, volume, etc.) through pixel analysis. However, each software has its own option and feature to obtain the particular information, for example, "MeasureObjectSizeAndShape" and "MeasureObjectIntensity" in CP and "Set measurement" and "ROI Manager" in ImageJ. Nonetheless, this further analysis is not within the scope of this article. We try to focus on the principle of image-based droplet detection in different software and their components that may ease the user with no experience in image-based analysis.

CONCLUSIONS

This investigation gives insights into processing droplet microfluidic images using the four currently most popular software tools. We classified the types of open-source software into rule-based and machine learning-based groups. Both groups have three levels of complexity that cover pre-processing, processing, and post-processing steps. These steps help users, specifically with no programming experience, to choose and perform their image analysis. In our experimental setup, we found that the rule-based type of software is better suited for image-based droplet detection. The rule-based type tools also have a simpler workflow or pipeline, especially aimed for non-experienced users. In our case, CP outperforms other software in terms of accuracy, precision, and user-friendliness (defined as usability for non-experienced users in building the pipeline and performing image-based droplet detection using available software modules). In terms of time processing, ImageJ and QuPath give faster processing time to detect droplets in 64 images. On the other hand, Ilastik gives a direct module that may ease early-stage researchers in image-based detection using the annotation principle. However, the optimal software choice may definitely be different for other users depending on their experimental conditions and acquired images. Our paper would serve as a starting point for them to compare available solutions and start with settings optimization, either using rule-based or machine learning-based software. In addition, published research, documentation, or forum discussions (such as www.image.sc) help in finding the most suitable software pipeline for image-based droplet detection and analysis.

METHODS

Software Search and Selection. We used selected software tools to detect droplets using the procedure explained by Bartkova et al.[31] We found several available and accessible software tools online such as CP,[35] ImageJ,[27] Ilastik,[36] QuPath,[37] Icy,[59] BioFilmQ,[60] CellOrganizer,[61] CellCognition,[62] BioImageXD,[63] BacStalk,[64] Advanced CellClassifier,[65] Phenoripper,[66] and Cytomine.[67] We have tested every software mentioned previously to perform image-based droplet detection; however, not all of the software had a good documentation, workflow, reference, and user-friendly interface. Therefore, we tried to find the most preferred tools available online by using Twint—Twitter Intelligence Tool script[34] written in Python and Scopus search from their website (https://www.scopus.com). The search has the same filter, including the search time (01-01-2010 until 31-12-2020), and only receives the result in "English". Therefore, the search both in Twitter and Scopus will not consider any data outside the filter. Both Twitter and Scopus data were obtained on February 11, 2021. We used each software's name as the keyword for the search. For the Twitter search, the processing was executed in Jupyter Notebook (ver. 6.0.3)[68] within Anaconda Navigator.[69] We also imported datetime and Pandas as additional libraries. For the Scopus search, it was performed using the same keyword. Both results were visualized together using Bokeh and NumPy libraries in Python.[70−72]

Droplet Generation and Image Acquisition. We repeated the method described in Bartkova et al. to generate droplets and their image data.[31] We used a set of 64 images to test the most popular software to detect droplets. The images are 2D layers of droplets generated by fluorescence confocal microscopy. We used the same images to find a suitable workflow for each software and describe it thoroughly in the next paragraph. Using the data, we calculated the precision and accuracy of detecting the droplets by comparing the results with manual counting using the same batch processing results in the same attempt.

Image Analysis with the Most Popular Software. The image data were analyzed first as a single image using ImageJ

https://doi.org/10.1021/acsomega.1c02664
ACS Omega 2021, 6, 22625−22634

(ver. 1.52p), CP (ver. 4.0.3), Ilastik (ver. 1.3.3), and QuPath (ver. 0.2.3). For each of the software, we describe the pipeline construction in the following paragraphs. Pipelines can be found at https://github.com/taltechmicrofluidics.

Pipeline Construction in CP. We used our previous pipeline[31] in CP as the basis for exploring other tools. We uploaded the image through a drag and drop feature in the Images module and set the Metadata, NamesAndTypes, and Groups according to our setting. We used the "IdentifyPrimaryObject" to detect droplets. We also used the same setting that is also provided in our GitHub repository (github.com/taltechmicrofluidics/CP-for-droplet-analysis). The "MeasureObjectIntensity" and "ExportToSpreadSheet" modules were also set as previously. The results were obtained automatically after pressing the "Analyze Image" button.

Pipeline Construction in ImageJ. For ImageJ, we recorded the workflow in the macro record option. This record was used to make scripts for batch processing. To upload the image, we use Open Image from the File tab in the main menu. The parameter was set within "Set Measurement" under "Analyze" tab, and we only ticked "Area" for obtaining the pixels' area in one droplet. This was followed with processing workflow, which included segmentation using "Threshold" under "Adjust" option in the "Image" tab. The threshold was determined as 1507, corresponding to 0.023 scale, described in our previous article using CP. The thresholding was followed with "Watershed" to separate droplets from each other. The counting was performed using "Analyze Particle" under the "Analyze" tab. We set the size corresponding to the range we described in CP, 22,500 up to 62,500 pixels^2 with 0 circularity. Once we finished the processing step, we downloaded the image through the "Flatten" option in the "ROI Manager" menu. We obtained the results in the table, which appeared straight after we performed the analysis.

Pipeline Construction in Ilastik. In Ilastik, we used "Pixel Classification" and "Object Classification" pre-defined workflow. We loaded the image in the Input Data module and selected the features for the training set. Since we did not have any reference regarding this type of workflow, we used the recommendation from image.sc forum, starting by adding 0.30, 1.00, and 3.50 sigma or scale corresponding to the selected features, e.g., Gaussian Filter, for color/intensity, edge, and texture. We trained the program to distinguish between the background (dark) and droplets using manual annotations/labels. For thresholding, we used the default smoothing value (1.0 and 1.0) with a 0.70 threshold. For the size filter, we put values that correspond to the settings in ImageJ, 22,500 for the minimum size and 62,500 for the maximum size. This was followed by using the standard object selection feature option and selecting the detected droplets in object classification as a sample. After finishing the setup, we obtained the results by exporting both object predictions and measured features.

Pipeline Construction in QuPath. In QuPath, we started the workflow by creating a project (Create Project) and uploading the image (Add Image). Once the selected image was ready, we performed annotations similar to Ilastik. This process aimed to distinguish the background and foreground (droplets). After annotating the image, we performed "Pixel Classification" using the artificial neural network (ANN_MLP) classifier with high (downsample = 4.0) resolution. For the features, the scales were 1.0, 2.0, and 4.0 for Gaussian gradient magnitude, Hessian determinant, and Hessian max eigenvalue, respectively. We created object detection for droplets and measured all detected droplets. We set a thick boundary class to make borders between each of the droplets. We saved the measurement data from the measurement menu.

Batch Processing from Each of the Software. In CP, we performed batch processing by loading the set of images in the Images module and run the "Analyze Images" button. For ImageJ, we executed batch processing using the "Batch Process" option under the "Process" tab. We used a recorded macro with some adjustments to execute the images in the Input folder. By processing the images through this option, we generated results directly to the Output folder. In Ilastik, we continued the batch processing straight after setting up the workflow. Similar to CP, we executed batch processing after uploading the images and only needed to press the "Process all images" button. In QuPath, we transformed the workflow from a single image into scripts to execute the batch processing. Since QuPath provides the script builder, we did not have to script by ourselves, and we could start batch processing by executing the script and ran it for the whole image set in the project. However, the image results from QuPath require additional script using Groovy. We managed to generate the results and you may find the script in our GitHub. We stored both single and batch processing pipelines from each of the software here: (github.com/taltechmicrofluidics/Software-Analysis).

Data Acquisition and Processing. We gathered all results and processed them in Microsoft Excel as follows. We tested the results with sensitivity and specificity tests and used manual counting as the reference.[42,73,74] We used these formulas for the test:

$$\text{FP Rate} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

$$\text{TP Rate} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

Where TP is the correct droplet Detection compared to ground truth, FP is the wrong detection (detecting background), FN is the wrong detection (software cannot recognize existed droplet), TN is the background (0), accuracy is the quality of correctness, and precision is the similarity upon repeatable counting.

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acsomega.1c02664.

Software settings, detailed diagnostic test results, image quality description, and step-by-step guideline and utilization from the tested software (PDF)

■ AUTHOR INFORMATION

Corresponding Author

Ott Scheler − *Department of Chemistry and Biotechnology, Tallinn University of Technology, 12618 Tallinn, Estonia;* orcid.org/0000-0002-8428-1350; Email: ott.scheler@taltech.ee

Authors

Immanuel Sanka − *Department of Chemistry and Biotechnology, Tallinn University of Technology, 12618 Tallinn, Estonia*

Simona Bartkova − *Department of Chemistry and Biotechnology, Tallinn University of Technology, 12618 Tallinn, Estonia*

Pille Pata − *Department of Chemistry and Biotechnology, Tallinn University of Technology, 12618 Tallinn, Estonia*

Olli-Pekka Smolander − *Department of Chemistry and Biotechnology, Tallinn University of Technology, 12618 Tallinn, Estonia;* orcid.org/0000-0002-6795-7734

Complete contact information is available at: https://pubs.acs.org/10.1021/acsomega.1c02664

Author Contributions

The manuscript was written through contributions from all authors. I.S., O.S., and O.-P.S. conceived the study. I.S. conducted the research and wrote the article with support from all the other authors. S.B. and P.P. were responsible for the microbiology part, droplet experiments, and microscopy imaging. All authors have given their approval to the final version of the manuscript.

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

The research was performed partially in the laboratory setup with the support from the TTU Development Program 2016−2022 (project no. 2014−2020.4.01.16.0032). We also acknowledge the Estonian Research Council grants MOBTP109, PRG620, and MOBJD556. Authors received help for early software screening from Matin Nuhamunada, Afif Pranaya Jati, and Ahmad Ardi in Universitas Gadjah Mada (Indonesia).

■ REFERENCES

(1) Ding, Y.; Howes, P. D.; Demello, A. J. Recent Advances in Droplet Microfluidics. *Anal. Chem.* **2020**, *92*, 132−149.

(2) Scheler, O.; Postek, W.; Garstecki, P. Recent Developments of Microfluidics as a Tool for Biotechnology and Microbiology. *Curr. Opin. Biotechnol.* **2019**, *55*, 60−67.

(3) Diefenbach, X. W.; Farasat, I.; Guetschow, E. D.; Welch, C. J.; Kennedy, R. T.; Sun, S.; Moore, J. C. Enabling Biocatalysis by High-Throughput Protein Engineering Using Droplet Microfluidics Coupled to Mass Spectrometry. *ACS Omega* **2018**, *3*, 1498−1508.

(4) Vasudevan, H. N.; Xu, P.; Servellita, V.; Miller, S.; Liu, L.; Gopez, A.; Chiu, C. Y.; Abate, A. R. Digital Droplet PCR Accurately Quantifies SARS-CoV-2 Viral Load from Crude Lysate without Nucleic Acid Purification. *Sci. Rep.* **2021**, *11*, 780.

(5) Kaminski, T. S.; Scheler, O.; Garstecki, P. Droplet Microfluidics for Microbiology: Techniques, Applications and Challenges. *Lab Chip* **2016**, *16*, 2168−2187.

(6) Du, W.-B.; Sun, M.; Gu, S.-Q.; Zhu, Y.; Fang, Q. Automated Microfluidic Screening Assay Platform Based on DropLab. *Anal. Chem.* **2010**, *82*, 9941−9947.

(7) Hindson, B. J.; Ness, K. D.; Masquelier, D. A.; Belgrader, P.; Heredia, N. J.; Makarewicz, A. J.; Bright, I. J.; Lucero, M. Y.; Hiddessen, A. L.; Legler, T. C.; Kitano, T. K.; Hodel, M. R.; Petersen, J. F.; Wyatt, P. W.; Steenblock, E. R.; Shah, P. H.; Bousse, L. J.; Troup, C. B.; Mellen, J. C.; Wittmann, D. K.; Erndt, N. G.; Cauley, T. H.; Koehler, R. T.; So, A. P.; Dube, S.; Rose, K. A.; Montesclaros, L.; Wang, S.; Stumbo, D. P.; Hodges, S. P.; Romine, S.; Milanovich, F. P.; White, H. E.; Regan, J. F.; Karlin-Neumann, G. A.; Hindson, C. M.; Saxonov, S.; Colston, B. W. High-Throughput Droplet Digital PCR System for Absolute Quantitation of DNA Copy Number. *Anal. Chem.* **2011**, *83*, 8604−8610.

(8) Hindson, C. M.; Chevillet, J. R.; Briggs, H. A.; Gallichotte, E. N.; Ruf, I. K.; Hindson, B. J.; Vessella, R. L.; Tewari, M. Absolute Quantification by Droplet Digital PCR versus Analog Real-Time PCR. *Nat. Methods* **2013**, *10*, 1003−1005.

(9) Scheler, O.; Pacocha, N.; Debski, P. R.; Ruszczak, A.; Kaminski, T. S.; Garstecki, P. Optimized Droplet Digital CFU Assay (DdCFU) Provides Precise Quantification of Bacteria over a Dynamic Range of 6 Logs and Beyond. *Lab Chip* **2017**, *17*, 1980−1987.

(10) Scheler, O.; Makuch, K.; Debski, P. R.; Horka, M.; Ruszczak, A.; Pacocha, N.; Sozański, K.; Smolander, O. P.; Postek, W.; Garstecki, P. Droplet-Based Digital Antibiotic Susceptibility Screen Reveals Single-Cell Clonal Heteroresistance in an Isogenic Bacterial Population. *Sci. Rep.* **2020**, *10*, 1−8.

(11) De Cesare, I.; Zamora-Chimal, C. G.; Postiglione, L.; Khazim, M.; Pedone, E.; Shannon, B.; Fiore, G.; Perrino, G.; Napolitano, S.; Di Bernardo, D.; Savery, N. J.; Grierson, C.; Di Bernardo, M.; Marucci, L. ChipSeg: An Automatic Tool to Segment Bacterial and Mammalian Cells Cultured in Microfluidic Devices. *ACS Omega* **2021**, *6*, 2473−2476.

(12) Zantow, M.; Dendere, R.; Douglas, T. S. Image-Based Analysis of Droplets in Microfluidics. In *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*; Institute of Electrical and Electronics Engineers, 2013; pp. 1776−1779.

(13) Zhu, Y.; Fang, Q. Analytical Detection Techniques for Droplet Microfluidics-A Review. *Anal. Chim. Acta* **2013**, *787*, 24−35.

(14) Harmon, J. B.; Gray, H. K.; Young, C. C.; Schwab, K. J. Microfluidic Droplet Application for Bacterial Surveillance in Fresh-Cut Produce Wash Waters. *PLoS One* **2020**, *15*, No. e0233239.

(15) Najah, M.; Mayot, E.; Mahendra-Wijaya, I. P.; Griffiths, A. D.; Ladame, S.; Drevelle, A. New Glycosidase Substrates for Droplet-Based Microfluidic Screening. *Anal. Chem.* **2013**, *85*, 9807−9814.

(16) Chen, P.; Chen, D.; Li, S.; Ou, X.; Liu, B.-F. Microfluidics towards Single Cell Resolution Protein Analysis. *TrAC, Trends Anal. Chem.* **2019**, 2−12.

(17) Cui, X.; Ren, L.; Shan, Y.; Wang, X.; Yang, Z.; Li, C.; Xu, J.; Ma, B. Smartphone-Based Rapid Quantification of Viable Bacteria by Single-Cell Microdroplet Turbidity Imaging. *Analyst* **2018**, *143*, 3309−3316.

(18) Samacoits, A.; Nimsamer, P.; Mayuramart, O.; Chantaravisoot, N.; Sitthi-Amorn, P.; Nakhakes, C.; Luangkamchorn, L.; Tongcham, P.; Zahm, U.; Suphanpayak, S.; Padungwattanachoke, N.; Leelarthaphin, N.; Huayhongthong, H.; Pisitkun, T.; Payungporn, S.; Hannanta-Anan, P. Machine Learning-Driven and Smartphone-Based Fluorescence Detection for CRISPR Diagnostic of SARS-CoV-2. *ACS Omega* **2021**, *6*, 2727−2733.

(19) Vaithiyanathan, M.; Safa, N.; Melvin, A. T. FluoroCellTrack: An Algorithm for Automated Analysis of High-Throughput Droplet Microfluidic Data. *PLoS One* **2019**, *14*, No. e0215337.

(20) Genot, A. J.; Baccouche, A.; Sieskind, R.; Aubert-Kato, N.; Bredeche, N.; Bartolo, J. F.; Taly, V.; Fujii, T.; Rondelez, Y. High-Resolution Mapping of Bifurcations in Nonlinear Biochemical Circuits. *Nat. Chem.* **2016**, *8*, 760−767.

(21) Baccouche, A.; Okumura, S.; Sieskind, R.; Henry, E.; Aubert-Kato, N.; Bredeche, N.; Bartolo, J.-F.; Taly, V.; Rondelez, Y.; Fujii, T.; Genot, A. J. Massively Parallel and Multiparameter Titration of Biochemical Assays with Droplet Microfluidics. *Nat. Protoc.* **2017**, *12*, 1912−1932.

(22) Svensson, C.-M.; Shvydkiv, O.; Dietrich, S.; Mahler, L.; Weber, T.; Choudhary, M.; Tovar, M.; Figge, M. T.; Roth, M. Coding of Experimental Conditions in Microfluidic Droplet Assays Using Colored Beads and Machine Learning Supported Image Analysis. *Small* **2018**, *15*, 1802384.

(23) Byrnes, S. A.; Phillips, E. A.; Huynh, T.; Weigl, B. H.; Nichols, K. P. Polydisperse Emulsion Digital Assay to Enhance Time to Detection and Extend Dynamic Range in Bacterial Cultures Enabled by a Statistical Framework. *Analyst* **2018**, *143*, 2828−2836.

(24) Dressler, O. J.; Howes, P. D.; Choo, J.; Demello, A. J. Reinforcement Learning for Dynamic Microfluidic Control. *ACS Omega* **2018**, *3*, 10084−10091.
(25) Zang, E.; Brandes, S.; Tovar, M.; Martin, K.; Mech, F.; Horbert, P.; Henkel, T.; Figge, M. T.; Roth, M. Real-Time Image Processing for Label-Free Enrichment of Actinobacteria Cultivated in Picolitre Droplets. *Lab Chip* **2013**, *13*, 3707−3713.
(26) Sarkar, S.; Cohen, N.; Sabhachandani, P.; Konry, T. Phenotypic Drug Profiling in Droplet Microfluidics for Better Targeting of Drug-Resistant Tumors. *Lab Chip* **2015**, *15*, 4441−4450.
(27) Schindelin, J.; Arganda-Carreras, I.; Frise, E.; Kaynig, V.; Longair, M.; Pietzsch, T.; Preibisch, S.; Rueden, C.; Saalfeld, S.; Schmid, B.; Tinevez, J. Y.; White, D. J.; Hartenstein, V.; Eliceiri, K.; Tomancak, P.; Cardona, A. Fiji: An Open-Source Platform for Biological-Image Analysis. *Nat. Methods* **2012**, *9*, 676−682.
(28) Najah, M.; Griffiths, A. D.; Ryckelynck, M. Teaching Single-Cell Digital Analysis Using Droplet-Based Microfluidics. *Anal. Chem.* **2012**, *84*, 1202−1209.
(29) Lamprecht, M. R.; Sabatini, D. M.; Carpenter, A. E. CellProfiler™: Free, Versatile Software for Automated Biological Image Analysis. *BioTechniques* **2007**, *42*, 71−75.
(30) Miura, K.; Paul-Gilloteaux, P.; Tosi, S.; Colombelli, J. Workflows and Components of Bioimage Analysis. In *Bioimage Data Analysis Workflows*; Springer: Cham, 2020; pp. 1−7.
(31) Bartkova, S.; Vendelin, M.; Sanka, I.; Pata, P.; Scheler, O. Droplet Image Analysis with User-Friendly Freeware CellProfiler. *Anal. Methods* **2020**, *12*, 2287−2294.
(32) Malik, A.; Heyman-Schrum, C.; Johri, A. Use of Twitter across Educational Settings: A Review of the Literature. *International Journal of Educational Technology in Higher Education*; Springer: Netherlands, 2019, pp. 1−22.
(33) Klar, S.; Krupnikov, Y.; Ryan, J. B.; Searles, K.; Shmargad, Y. Using Social Media to Promote Academic Research: Identifying the Benefits of Twitter for Sharing Academic Work. *PLoS One* **2020**, *15*, No. e0229446.
(34) OSIN team - *Twint Project. GitHub - Twintproject/Twint: An Advanced Twitter Scraping & OSINT Tool Written in Python That Doesn't Use Twitter's API, Allowing You to Scrape a User's Followers, Following, Tweets and More While Evading Most API Limitations*. 2020, p https://github.com/twintproject/twint.
(35) McQuin, C.; Goodman, A.; Chernyshev, V.; Kamentsky, L.; Cimini, B. A.; Karhohs, K. W.; Doan, M.; Ding, L.; Rafelski, S. M.; Thirstrup, D.; Wiegraebe, W.; Singh, S.; Becker, T.; Caicedo, J. C.; Carpenter, A. E. CellProfiler 3.0: Next-Generation Image Processing for Biology. *PLoS Biol.* **2018**, *16*, No. e2005970.
(36) Berg, S.; Kutra, D.; Kroeger, T.; Straehle, C. N.; Kausler, B. X.; Haubold, C.; Schiegg, M.; Ales, J.; Beier, T.; Rudy, M.; Eren, K.; Cervantes, J. I.; Xu, B.; Beuttenmueller, F.; Wolny, A.; Zhang, C.; Koethe, U.; Hamprecht, F. A.; Kreshuk, A. Ilastik: Interactive Machine Learning for (Bio)Image Analysis. *Nat. Methods* **2019**, *16*, 1226−1232.
(37) Bankhead, P.; Loughrey, M. B.; Fernández, J. A.; Dombrowski, Y.; McArt, D. G.; Dunne, P. D.; McQuaid, S.; Gray, R. T.; Murray, L. J.; Coleman, H. G.; James, J. A.; Salto-Tellez, M.; Hamilton, P. W. QuPath: Open Source Software for Digital Pathology Image Analysis. *Sci. Rep.* **2017**, *7*, 16878.
(38) Liang, Y.; Zhang, M.; Browne, W. N. Image Segmentation: A Survey of Methods Based on Evolutionary Computation. In *Simulated Evolution and Learning. SEAL 2e014. Lecture Notes in Computer Science*; Springer: Verlag, 2014; Vol. *8886*, pp. 847−859.
(39) Uchida, S. Image Processing and Recognition for Biological Images. *Dev., Growth Differ.* **2013**, *55*, 523−549.
(40) Darvishan, A.; Bakhshi, H.; Madadkhani, M.; Mir, M.; Bemani, A. Application of MLP-ANN as a Novel Predictive Method for Prediction of the Higher Heating Value of Biomass in Terms of Ultimate Analysis. *Energy Sources, Part A* **2018**, *40*, 2960−2966.
(41) Abdul-Jawad, B. *Groovy and Grails Recipes*; Apress Media: LLC, 2009.
(42) Gaddis, G. M.; Gaddis, M. L. Introduction to Biostatistics: Part 3, Sensitivity, Specificity, Predictive Value, and Hypothesis Testing. *Ann. Emerg. Med.* **1990**, *19*, 591−597.
(43) Cecconi, M.; Rhodes, A.; Poloniecki, J.; Della Rocca, G.; Grounds, R. M. Bench-to-Bedside Review: The Importance of the Precision of the Reference Technique in Method Comparison Studies–with Specific Reference to the Measurement of Cardiac Output. *Crit. Care* **2009**, *13*, 201.
(44) Bland, J. M.; Altman, D. G. Comparing Methods of Measurement: Why Plotting Difference against Standard Method Is Misleading. *Lancet* **1995**, *346*, 1085−1087.
(45) Bland, J. M.; Altman, D. G. Measuring Agreement in Method Comparison Studies. *Stat. Methods Med. Res.* **1999**, *8*, 135−160.
(46) Carpenter, A. E.; Jones, T. R.; Lamprecht, M. R.; Clarke, C.; Kang, I.; Friman, O.; Guertin, D. A.; Chang, J.; Lindquist, R. A.; Moffat, J.; Golland, P.; Sabatini, D. M. CellProfiler: Image Analysis Software for Identifying and Quantifying Cell Phenotypes. *Genome Biol.* **2006**, *7*, R100.
(47) Frank, M.; Drikakis, D.; Charissis, V. Machine-Learning Methods for Computational Science and Engineering. *Computation* **2020**, *8*, 15.
(48) Ribeiro, G. P.; Endringer, D. C.; De Andrade, T. U.; Lenz, D. Comparison between Two Programs for Image Analysis, Machine Learning and Subsequent Classification. *Tissue Cell* **2019**, *58*, 12−16.
(49) Bonnal, R. J. P.; Yates, A.; Goto, N.; Gautier, L.; Willis, S.; Fields, C.; Katayama, T.; Prins, P. Sharing Programming Resources between Bio* Projects. In *Methods in Molecular Biology*; Humana Press Inc., 2019; Vol. *1910*, pp. 747−766.
(50) Fourment, M.; Gillings, M. R. A Comparison of Common Programming Languages Used in Bioinformatics. *BMC Bioinformatics* **2008**, *9*, 82.
(51) Schneider, C. A.; Rasband, W. S.; Eliceiri, K. W. NIH Image to ImageJ: 25 Years of Image Analysis. *Nat. Methods* **2012**, *9*, 671−675.
(52) Arganda-Carreras, I.; Kaynig, V.; Rueden, C.; Eliceiri, K. W.; Schindelin, J.; Cardona, A.; Sebastian Seung, H. Trainable Weka Segmentation: A Machine Learning Tool for Microscopy Pixel Classification. *Bioinformatics* **2017**, *33*, 2424−2426.
(53) Zanotelli, V. R.; Leutenegger, M.; Lun, X.; Georgi, F.; de Souza, N.; Bodenmiller, B. A Quantitative Analysis of the Interplay of Environment, Neighborhood, and Cell State in 3D Spheroids. *Mol. Syst. Biol.* **2020**, *16* (), DOI: 10.15252/msb.20209798.
(54) Jones, T. R.; Kang, I. H.; Wheeler, D. B.; Lindquist, R. A.; Papallo, A.; Sabatini, D. M.; Golland, P.; Carpenter, A. E. CellProfiler Analyst: Data Exploration and Analysis Software for Complex Image-Based Screens. *BMC Bioinformatics* **2008**, *9*, 482.
(55) Dao, D.; Fraser, A. N.; Hung, J.; Ljosa, V.; Singh, S.; Carpenter, A. E. CellProfiler Analyst: Interactive Data Exploration, Analysis and Classification of Large Biological Image Sets. *Bioinformatics* **2016**, *32*, 3210−3212.
(56) Ochoa, A.; Álvarez-Bohórquez, E.; Castillero, E.; Olguin, L. F. Detection of Enzyme Inhibitors in Crude Natural Extracts Using Droplet-Based Microfluidics Coupled to HPLC. *Anal. Chem.* **2017**, *89*, 4889−4896.
(57) Mary, P.; Studer, V.; Tabeling, P. Microfluidic Droplet-Based Liquid-Liquid Extraction. *Anal. Chem.* **2008**, *80*, 2680−2687.
(58) Tamminen, J.; Lahdenperä, E.; Koiranen, T.; Kuronen, T.; Eerola, T.; Lensu, L.; Kälviäinen, H. Determination of Single Droplet Sizes, Velocities and Concentrations with Image Analysis for Reactive Extraction of Copper. *Chem. Eng. Sci.* **2017**, *167*, 54−65.
(59) De Chaumont, F.; Dallongeville, S.; Olivo-Marin, J. C. ICY: A New Open-Source Community Image Processing Software. In *International Symposium on Biomedical Imaging*; IEEE Computer Society, 2011; pp. 234−237.
(60) Hartmann, R.; Jeckel, H.; Jelli, E.; Singh, P. K.; Vaidya, S.; Bayer, M.; Rode, D. K. H.; Vidakovic, L.; Díaz-Pascual, F.; Fong, J. C. N.; Dragoš, A.; Lamprecht, O.; Thöming, J. G.; Netter, N.; Häussler, S.; Nadell, C. D.; Sourjik, V.; Kovács, Á. T.; Yildiz, F. H.; Drescher, K. Quantitative Image Analysis of Microbial Communities with BiofilmQ. *Nat. Microbiol.* **2021**, *6*, 151−156.

(61) Majarian, T. D.; Cao-Berg, I.; Ruan, X.; Murphy, R. F. CellOrganizer: Learning and Using Cell Geometries for Spatial Cell Simulations. In *Methods in Molecular Biology*; Humana Press Inc., 2019; Vol. *1945*, pp. 251−264.
(62) Held, M.; Schmitz, M. H. A.; Fischer, B.; Walter, T.; Neumann, B.; Olma, M. H.; Peter, M.; Ellenberg, J.; Gerlich, D. W. CellCognition: Time-Resolved Phenotype Annotation in High-Throughput Live Cell Imaging. *Nat. Methods* **2010**, *7*, 747−754.
(63) Kankaanpää, P.; Paavolainen, L.; Tiitta, S.; Karjalainen, M.; Päivärinne, J.; Nieminen, J.; Marjomäki, V.; Heino, J.; White, D. J. BioImageXD: An Open, General-Purpose and High-Throughput Image-Processing Platform. *Nat. Methods* **2012**, *9*, 683−689.
(64) Hartmann, R.; Teeseling, M. C. F.; Thanbichler, M.; Drescher, K. BacStalk: A Comprehensive and Interactive Image Analysis Software Tool for Bacterial Cell Biology. *Mol. Microbiol.* **2020**, *114*, 140−150.
(65) Piccinini, F.; Balassa, T.; Szkalisity, A.; Molnar, C.; Paavolainen, L.; Kujala, K.; Buzas, K.; Sarazova, M.; Pietiainen, V.; Kutay, U.; Smith, K.; Horvath, P. Advanced Cell Classifier: User-Friendly Machine-Learning-Based Software for Discovering Phenotypes in High-Content Imaging Data. *Cell Syst.* **2017**, *4*, 651−655.e5.
(66) Rajaram, S.; Pavie, B.; Wu, L. F.; Altschuler, S. J. PhenoRipper: Software for Rapidly Profiling Microscopy Images. *Nat. Methods* **2012**, *9*, 635−637.
(67) Marée, R.; Rollus, L.; Stévens, B.; Hoyoux, R.; Louppe, G.; Vandaele, R.; Begon, J. M.; Kainz, P.; Geurts, P.; Wehenkel, L. Collaborative Analysis of Multi-Gigapixel Imaging Data Using Cytomine. *Bioinformatics* **2016**, *32*, 1395−1401.
(68) Kluyver, T.; Ragan-Kelley, B.; Pérez, F.; Granger, B.; Bussonnier, M.; Frederic, J.; Kelley, K.; Hamrick, J.; Grout, J.; Corlay, S.; Ivanov, P.; Avila, D.; Abdalla, S.; Willing, C. Jupyter Notebooks—a Publishing Format for Reproducible Computational Workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas - Proceedings of the 20th International Conference on Electronic Publishing, ELPUB 2016*; IOS Press: BV, 2016; pp. 87−90.
(69) *Anaconda Software Distribution. Anaconda Software Distribution*. Anaconda 2016, p https://www.anaconda.com/.
(70) The Pandas Development Team. *Pandas-Dev/Pandas: Pandas 1.2.3. Zenodo* 2020, p https://zenodo.org/record/4572994, DOI: 10.5281/ZENODO.4572994.
(71) Bokeh Development Team. *Bokeh: Python Library for Interactive Visualization*. 2018, p http://www.bokeh.pydata.org.
(72) Harris, C. R.; Millman, K. J.; van der Walt, S. J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N. J.; Kern, R.; Picus, M.; Hoyer, S.; van Kerkwijk, M. H.; Brett, M.; Haldane, A.; del Río, J. F.; Wiebe, M.; Peterson, P.; Gérard-Marchant, P.; Sheppard, K.; Reddy, T.; Weckesser, W.; Abbasi, H.; Gohlke, C.; Oliphant, T. E. Array Programming with NumPy. *Nature* **2020**, *585*, 357−362.
(73) Soleymani, R.; Granger, E.; Fumera, G. F-Measure Curves: A Tool to Visualize Classifier Performance under Imbalance. *Pattern Recognit.* **2020**, *100*, 107146.
(74) Landgrebe, T. C. W.; Paclik, P.; Duin, R. P. W.; Bradley, A. P. Precision-Recall Operating Characteristic (P-ROC) Curves in Imprecise Environments. In *Proceedings - International Conference on Pattern Recognition*; Institute of Electrical and Electronics Engineers, 2006; Vol. *4*, pp. 123−127.

https://doi.org/10.1021/acsomega.1c02664
ACS Omega 2021, 6, 22625−22634