pyPOCQuant — A tool to automatically quantify Point-Of-Care Tests from images

Journal Article

Author(s):

Cuny, Andreas P.; Rudolf, Fabian; [Ponti, Aaron](#)

Publication date:

2021-07

Permanent link:

<https://doi.org/10.3929/ethz-b-000491291>

Rights / license:

[Creative Commons Attribution 4.0 International](#)

Originally published in:

SoftwareX 15, <https://doi.org/10.1016/j.softx.2021.100710>

Original software publication

pyPOCQuant — A tool to automatically quantify Point-Of-Care Tests from images

Andreas P. Cuny^{a,b}, Fabian Rudolf^{a,b}, Aaron Ponti^{a,*}^a ETH Zurich, Department of Biosystems Science and Engineering, Mattenstr. 26, 4058 Basel, Switzerland^b Swiss Institute of Bioinformatics, Mattenstr. 26, 4058 Basel, Switzerland

ARTICLE INFO

Article history:

Received 12 January 2021

Received in revised form 19 April 2021

Accepted 13 May 2021

Keywords:

Lateral flow assay (LFA)

Lateral flow immunoassay (LFIA)

Point of care test (POCT)

Test line quantification

Readout zone quantification

Diagnostics

Computer vision

QR code

Rapid testing

Rapid diagnostic tests (RTD)

ABSTRACT

Lateral flow Point-Of-Care Tests (POCTs) are a valuable tool for rapidly detecting pathogens and the associated immune response in humans and animals. In the context of the SARS-CoV-2 pandemic, they offer rapid on-site diagnostics and can relieve centralized laboratory testing sites, thus freeing resources that can be focused on especially vulnerable groups. However, visual interpretation of the POCT test lines is subjective, error prone and only qualitative. Here we present pyPOCQuant, an open-source tool implemented in Python 3 that can robustly and reproducibly analyze POCTs from digital images and return an unbiased and quantitative measurement of the POCT test lines.

© 2021 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

Code metadata

Current code version	v1.1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-21-00010
Code Ocean compute capsule	–
Legal Code License	GPL v3
Code versioning system used	git
Software code languages, tools, and services used	Python
Compilation requirements, operating environments & dependencies	see QuickStart manual
If available Link to developer documentation/manual	https://pypocquant.readthedocs.io/en/latest/index.html
Support email for questions	andreas.cuny@bsse.ethz.ch or aaron.ponti@bsse.ethz.ch

Software metadata

Current software version	1.1.0
Permanent link to executables of this version	https://gitlab.com/csb.ethz/pypocquantui/-/tags/1.1.0
Legal Software License	GPL v3
Computing platforms/Operating Systems	Microsoft Windows, Mac, Linux
Installation requirements & dependencies	see QuickStart manual
If available, link to user manual — if formally published include a reference to the publication in the reference list	https://pypocquant.readthedocs.io/en/latest/index.html
Support email for questions	same as above

1. Motivation and significance

Point of care tests (POCT) and rapid diagnostic tests (RTD) are a cost-effective and instrument-free alternative to molecular laboratory testing used for a wide variety of diseases and the

* Corresponding author.

E-mail address: aaron.ponti@bsse.ethz.ch (Aaron Ponti).

associated immune response, especially in times or regions where resources are limited [1]. Most POCTs are Immunochromatographic Lateral Flow Assays (LFA) or similar, that use one binding protein coated on a membrane and a second protein fused to a colloidal Gold entity to report a positive reaction as a visible color line [2]. The main components of an LFA (Fig. 1a, b) are, therefore, a nitrocellulose membrane with test and control lines, a sample pad where a conjugate is stored, and the absorption pad [3].

POCTs are commonly used for Streptococcus diagnostics and have been used to combat the epidemics of HIV, TB, hepatitis, dengue fever, and malaria. They all use binding proteins specific for the pathogen as their test line and color conjugate. Analogously, antibody assays reveal whether a patient's immune system has already developed antibodies (i.e., IgM, IgA, or IgG) against past viral infections. A sample is taken using a swab (e.g., in Streptococcus diagnostic), a small amount of blood (e.g., in antibody detection) or similar, and mixed with running buffer either outside or directly on the sample pad of the POCT. In the best case, POCTs deliver fast readouts (typically within 15 min) close to the patient and allow for fast reactions with a potential advantage for the patient and the whole local community [4].

To be of use in the global SARS-CoV-2 pandemic, POCTs must fulfill the standards set by the WHO ($\geq 80\%$ sensitivity and $\geq 97 - 99.5\%$ specificity) [5,6]. These specifications hold both for the diagnostic of an ongoing viral infection and the detection of past infections. A plethora of manufacturers provide tests that they claim to fulfill these specifications, but we argue that they will require comprehensive characterization and comparison of their analytical and clinical performance [7,8].

LFA and similar devices are scored visually and yield a dichotomous (positive/negative) readout. Experimentally, the analytical limit of detection can be determined using a titration of (repeated) measurements of purified proteins (such as monoclonal antibodies as a surrogate for blood IgG detection) or in proteins from the pathogen of interest that are expressed in cell lines and purified (i.e. Spike protein of SARS-CoV-2) [9]. However, they do have the potential for a quantitative readout using dedicated readers, lab scanner, or image-based analysis [10–13]. While there have been efforts to use smartphones as readers in combination with 3D-printed adapters and holders [14–20], no freely available pipeline is available to analyze POCTs from large numbers of images in an automated way.

Here, we present pyPOCQuant, an open-source tool for the large-scale, quantitative and automated analysis of large numbers of POCTs from a variety of different manufacturers independent of design. pyPOCQuant processes images of POCTs taken on digital cameras and returns both a binary result (positive vs. negative) for the test and control lines, and a quantitative readout of each test line. These robust and reproducible readouts are well suited for studying the temporal evolution of the patient's immune response to the infectious agent, or as a quick diagnostics tool for detecting a pathogen. As such, pyPOCQuant helps with the validation and characterization of different POCTs, but it can also make the use of these assays more reliable.

2. Software description

pyPOCQuant is a cross platform tool written in Python 3 and implemented as a stand-alone computer vision library. It is available under the open-source license GPLv3 on GitHub, both as source or as compiled stand-alone executable. Extensive documentation (including example scripts and data) can be found on readthedocs.org and bundled with the application. A QuickStart guide highlights the steps required to properly set up image acquisition and to run a successful analysis. The user manual describes its usage in full detail.

2.1. Software architecture

Depending on the workflow, needs and preferences of the user, pyPOCQuant can be launched from the command line interface (CLI) using the `pyPOCQuant.py` main script, or run as a module from a Jupyter notebook. We also developed pyPOCQuantUI, a user-friendly graphical user interface (GUI) and desktop application that leverages pyPOCQuant while massively facilitating the definition of essential parameters needed for the analysis. pyPOCQuantUI is implemented in Python 3 using the PyQt5 library and can be run from source or as a stand-alone, executable desktop application.

Fig. 2 displays a flowchart of the operations performed by pyPOCQuant. In short, for every image in a folder, pyPOCQuant extracts patient and metadata information, segments and processes the POCT, extracts its sensor area, locates and quantifies the test-line signals. All results are stored in a data frame that can be used for further statistical analysis. The next sections will discuss these operations in detail.

2.2. Software functionalities

The core of pyPOCQuant is a pipeline of operations that can be configured and started either from the CLI or GUI. Parameters are passed to the pipeline via a configuration file.

2.2.1. Pipeline

The pipeline processes all images in a user-defined input folder in parallel, and the number of parallel tasks can be defined at run time (Fig. 2). Results are compiled in a common data frame and saved as a comma-separated-value file in a user-defined output folder. The following sections describe the operations that are performed on each image independently.

Metadata retrieval. Current image from a series of formats (JPEG and various RAW image formats, specifically Nikon NEF, Sony ARW and Canon CR2) is read from disk. Image metadata such as acquisition date and time, exposure time, focal length, ISO, are read and stored in the data frame. Use of the provided template (Suppl. Fig. S6) for image acquisition ensures that patient information stored in a QR code placed on the template can be automatically retrieved and associated directly to the image.

POCT segmentation. Additional QR codes encoding spatial information make the process of segmentation more robust. These QR codes ensure that the image was acquired in the expected orientation. If this is not the case, a coarse initial rotation (in steps of 90 degrees) is performed. The positional QR codes delimit the area where the POCT is placed on the template (Suppl. Fig. S6). The POCT area is large enough to guarantee that POCTs of different shapes and sizes can be contained and robustly identified. The POCT area and later the POCT itself are extracted and finely rotated to be parallel to the image axes.

Readout zone (sensor) segmentation. Once the properly aligned POCT area has been segmented from the image, a candidate sensor window of size `sensor_search_area` and centered at the `sensor_center` coordinates is extracted. Using the control line as a visual anchor and its expected position within the sensor area as defined in `peak_expected_relative_location`, the search area is properly centered around the `sensor_center` coordinates and the sensor of given `sensor_size` is cropped for further analysis.

Fig. 1. Typical design of lateral flow based POCTs. (a) Configuration of LFA. (b) POCT schematic enclosing the LFA. (c) Example QR code encoding patient metadata (details in text) with POCT area and POCT positioning on template (Suppl. Fig. S6).

Fig. 2. pyPOCQuant architecture overview. Bold arrows depict the main flow through the tool using the GUI. Regular arrows depict the CLI usage and optional ways of interaction.

Test lines (TL) quantification. As the first step in the detection and analysis of the TL signals, a one-dimensional signal profile is calculated as the median intensity along the short side of the sensor window. This increases the robustness of the analysis by reducing variations in the TL signals due to inhomogeneous reactions of the bands (Fig. 3). Also, since the edges of the readout zone often show some impurities, *sensor_border* rows and columns of the sensor image are ignored when calculating the signal profile.

A robust linear regressor (Huber regressor, [21]) is used to fit the background level, that often shows an increasing or decreasing trend along the profile. Local minima in the profile are iteratively used to build a background line that is subtracted from the profile. A small constant subtracted from the background line guarantees that no values of the subtracted profile are negative. Next, the profile is low-pass filtered with a small Gaussian kernel ($\sigma = 1$) to clean spurious signal variations, and a local max detector tuned on the size of the expected TLs provides candidate location for the test lines. To avoid scoring local maxima due to noisy signal variations, a thresholding step is performed to make sure that only significant peaks are considered for further analysis. This threshold is calculated from the median variation of the background local minima and

the multiplicative factor *sensor_thresh_factor*. The remaining local maxima are scored by their relative proximity to the *peak_expected_relative_location* and assigned to one of the T1, T2 or Ct1 TLs. Last, each TL signal is calculated as the integral of the profile from background level to background level on the two sides of the peak. This decision was taken after observing that a binary segmentation of the TLs could result in a significant variation in the extent of the TLs across images, and thus increase the error rate in the detection of absolute and relative signal strengths. Finally, specifying the expected positions of the TLs and not simply their number, the TL assignment step becomes more robust against deviations in the expected position of the sensor, especially when only a subset of TLs is visible.

Once all TLs have been extracted and their absolute signals calculated, the binary readout is recorded. A TL is present if a significant peak could be extracted and its relative signal strength is normalized against the absolute signal of the control line. Absolute values are also reported in the results data frame to make sure that intra-batch variability can be accounted and corrected for. If the expected control line cannot be found and extracted from the sensor area, the result is not trusted: the test is recorded but is labeled as invalid.

Fig. 3. Test line quantification. (a) Background estimation (Est. bkgnd) and fixed offset (Bkgnd offset) from raw median intensity profile. (b) Background corrected intensity profile (Corr. profile) with peak maxima (red squares) and TL bounds (BD: blue, orange, green circle and lines) as well as peak significance threshold (red dashed line) and the estimated background (black dashed line). The raw gray-scale image is plotted below. The colored squares depict all the pixels considered for the signal quantification. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Outputs. The results for each image are collected into a data frame and saved into the output folder. The structure of the results file is described in the user manual. Result images and a full log of the process are saved along. The verbosity of the log and the creation of additional quality control images can be controlled in the settings.

2.2.2. Patient sample QR code label generator

When working with many POCTs from many patients and/or replicates, it is essential to link samples and patients unequivocally. Typically, an identification number is associated with each sample. To facilitate the association of patient and test metadata with the corresponding POCT analysis readouts, we have added a QR-code generation tool to pyPOCQuantUI that allows flexible printing of labels in the format SAMPLEID-MANUFACTURER-PLATE-WELL-USERDATA. The label can contain additional information such as sample id, manufacturer names, identification for sample replicates, organization in standardized plates (especially 96-well plates) and a short free text. With the default page and label settings the full label should not exceed 60 characters. The generated labels are compiled in one PDF file and can be printed with ordinary printers and placed on the template next to the POCT (Fig. 1c). pyPOCQuant will read this information and add it to the corresponding analysis results in the data frame.

3. Illustrative examples

3.1. Example usage with the GUI

pyPOCQuantUI (Fig. 4) facilitates the definition of the analysis parameters by offering interactive tools and visual cues. Also, its immediate feedback allows checking results and fine-tuning parameters quickly. The user manual describes all functionalities of the interface in full detail.

Configuration and working folders. pyPOCQuantUI starts with a default configuration already loaded. Alternatively, configuration files can be loaded from the file menu, the toolbar, or simply clicking on a *.conf file in the input file list. A new project is started by selecting an image folder (1). By default, the target results folder will target a pipeline subfolder in the input folder, but it can be changed (2). The files contained in the input folder are listed in (3).

Definition of basic parameters. Selecting one of the images in the list will display it on the right (4) and trigger the POCT extraction. Its progress is shown in (5). The extracted POCT is displayed in (6). By clicking on the draw sensor icon in the toolbar, the user can draw a polygon that connects the four corners of the readout zone of the POCT (7). This interactive sensor can be edited and deleted. Mouse wheel zooming or using the zoom icons facilitate the accurate drawing of the interactive sensor. The sensor polygon also features three vertical test lines that need to be aligned with the test lines to defined their expected relative position. The vertical lines' position can be changed with the mouse or by editing their values in the parameter tree (8).

Test and run pipeline. At this stage, all the basic parameters are set and can be tested by hitting the Test parameters button (9). It is recommended (and activated by default) to enable creation of quality control images and verbose logging during testing. When the parameters deliver satisfying results, one can increase the number of parallel workers (10) and start batch-processing all images in the input folder in parallel (11). The quantification_data.csv result file, log files, and, optionally, the quality control images will be stored in the output folder.

Advanced topics. pyPOCQuant's advanced parameters and detailed discussion of the results can be found in the User manual or in the QuickStart guide, which also summarizes a few common issues and suggests ways to resolve them. Worth mentioning is the Issue column in the result data frame that contains a non-zero error code if the analysis of the corresponding image could not be completed successfully. The issue codes can be found in the manual.

3.2. Example usage with the CLI

Running pyPOCQuant from the CLI is best suited when automating the processing of large amounts of images and folders. A configuration with default values is created (Listing 1) and the parameters are modified appropriately. The easiest approach is to use pyPOCQuantUI for this purpose, but it could also be done with other tools, such as Fiji (as described in the manual). To create a default configuration from the CLI, use the '-c' flag of pyPOCQuant.py.

Fig. 4. pyPOCQuantUI (GUI): interactively defining basic analysis parameters. The numbers correspond to the text.

Listing 1: Create a new default settings file

```
python pyPOCQuant.py -c /PATH/TO/CONFIG/FILE.conf
```

Once the configuration file is ready, a full study can be started by running pyPOCQuant on a full folder of images (Listing 2). The analysis is performed in parallel, and the number of concurrent tasks can be defined by the '-w' (--workers) argument. The maximum number of workers allowed corresponds to the maximum number of CPU cores.

Listing 2: Usage of pyPOCQuant with the CLI on an image folder with given settings file and number of workers

```
python pyPOCQuant.py -f /PATH/TO/INPUT/FOLDER -o /PATH/TO/RESULTS/FOLDER -s /PATH/TO/CONFIG/FILE -w {NUM_WORKERS}
```

The argument '-f' is the path to the folder that contains all images for a given camera and manufacturer to be processed; '-o' is the path where the 'quantification_data.csv' result file, log and quality control images are saved (optional, if omitted the path defaults to a pipeline sub-folder in the input folder); '-s' sets the path to the configuration file to be used for this analysis; '-w' is the number of parallel processes; e.g. 8.

3.3. Example usage with jupyter notebook

Finally, pyPOCQuant can be run from a Jupyter notebook. A full demonstration notebook along with example images can be found in the repository and the online documentation. Documentation of the parameters and outputs can be found in the User manual. Importing of run_pipeline and default_settings along with the path to images are enough to run pyPOCQuant from a Jupyter cell (Listing 3).

Listing 3: Minimal example of usage of pyPOCQuant in a Jupyter notebook

```
from pypocquant.lib.pipeline import run_pipeline
from pypocquant.lib.settings import default_settings
```

Get the default settings

```
settings = default_settings()
```

Change as needed

```
settings["sensor_band_names"] = ('igm', 'igg', 'ctl')
```

Alternatively, load existing settings

```
# from pypocquant.lib.settings import load_settings
# settings = load_settings('full/path/to/settings/file.conf')
```

Set final arguments

```
input_folder_path = 'full/path/to/input/folder'
results_folder_path = 'full/path/to/results/folder'
max_workers = 8
```

Run the pipeline

```
run_pipeline(
    input_folder_path,
    results_folder_path,
    **settings,
    max_workers=max_workers
)
```

3.4. Example results

As an illustration of the performance of pyPOCQuant, we analyzed $n=601$ images picked from studies performed on POCTs from 14 different manufacturers. The manufacturers considered are listed in Fig. 5. From each manufacturer we analyzed 50 images at random (with the exception of LUNGENE, ONESTEP, and PROGNOSIS, where we only had 12, 32, and 8 POCTs, respectively) [22]. Notice that several different camera models were used to acquire the images. For each manufacturer, one set of settings was defined and applied to all images in the corresponding set. The sensor_threshold_factor parameter was set to 3.0 throughout. Ground truth generated by human observers defined a grand total of 1745 positive TLs. Of those, pyPOCQuant correctly identified 1740 (1012 true positives TP, 728 true negatives TN, 4 false positives FP, and 1 false negative FN), resulting in an overall

Fig. 5. pyPOCQuant example results for 14 different POCT manufacturers. One image for each set ($n=50$) is shown. Top row depicts the background corrected signal and a red square marks the identified TLs. Middle row depicts the detected sensor. Bottom row depicts the POCT inside the raw QR code box. The third entry in the first row shows an example of a POCT that was wrongly positioned in the QR code box and that is ultimately rotated for sensor extraction (see supplementary material for a discussion of corrections). The table shows the pyPOCQuant results compared to a manual ground truth for each TL. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

accuracy of 0.997. In Fig. 5, the accuracy is shown as labeled dots (blue for TL1, orange for TL2 and green for CTL), with the global accuracy shown in black. A dashed line indicates the 0.95 accuracy level.

In the Supplementary Material we present a series of additional validation experiments and a preliminary quantitative analysis that suggests possible future applications of pyPOCQuant.

4. Impact

We developed pyPOCQuant to analyze POCT images as a rapid and reliable alternative to visual scoring of these devices. While humans can quickly and easily spot the presence or absence of TLs of POCTs, this assessment is always subjective and poorly reproducible. Different operators may easily interpret weak test line signals differently and even the same operator may score the same weak test line as positive or negative under different conditions. Therefore, human assessment is neither reliable nor quantitative.

We propose pyPOCQuant as a valuable tool that generates quantitative and reproducible POCT results from lab evaluations all the way to large-scale testing campaigns. pyPOCQuant is not dependent on specialized equipment and can be used with POCTs from many different manufacturers. It therefore allows for straightforward quantification of cheap, mass-producible and readily available POCT using a consumer (or even a smartphone) camera and a mid-end computer.

The efficient scoring of POCTs by pyPOCQuant allows for simple validation, and improves their reliability in every day use. Scaling up is feasible as it can take advantage of multi-core computers to process images in parallel making it suited for large studies with thousands of associated images. As such, pyPOCQuant can help when large-scale testing using classic approaches such as PCR or ELISA may be hampered by limited access to reagents and laboratory equipment, such as during the global SARS-CoV-2 pandemic.

5. Conclusions

pyPOCQuant delivers accurate detection and precise quantification of test and control lines on any LFA. pyPOCQuant is available as a library, command-line tool and graphical user-interface to fit into various analysis workflows. We developed it to reproducibly batch-process thousands of images of POCTs in parallel using multi-core computers or even clusters. We made pyPOCQuant user-friendly by adding a graphical user interface to facilitate loading images, defining and testing analysis parameters on single images, and launching entire analysis campaigns with a few clicks. In addition to its execution from the CLI or GUI, pyPOCQuant can also run in Jupyter notebooks. Notebooks integrate analysis and documentation into elegant reports.

Image acquisition requirements are undemanding: consumer cameras provide acceptable images for analysis (6–12 MPixel cameras offer a good compromise between image resolution and size), and the provided imaging template makes pyPOCQuant compatible with a wide variety of POCT housing designs and sizes.

pyPOCQuant helps validating POCTs quickly and can be used to establish rapid field testing. It has already been used in several pilot SARS-CoV-2 related projects involving several thousand images. Since POCTs are not restricted to SARS-CoV-2 antigens, pyPOCQuant can be broadly used for large-scale diagnostics and studies of various other diseases.

Future work may move beyond binary characterization of the POCTs (i.e., positive vs. negative readout) to a detailed quantitative analysis of the test lines. For instance, calibration of the pyPOCQuant response against samples of known concentration would allow pyPOCQuant to estimate the absolute concentration of the detected antigen or antibody in a POCT.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We would like to thank Marvin Wyss and Talea Marty from the Fachhochschule Nordwestschweiz for valuable user feedback that helped us greatly improve the software and the user experience. We thank Peter Spiess from the Fachhochschule Nordwestschweiz who provided images for the analysis of Figures S1 and S3. Marvin Wyss provided data for the quantitative validation of pyPOCQuant in Figure S4 and for part of the analysis of Fig. 5. Finally, Anna Stürmann, Beatrice Nickel, and Marie-Therese Ruf from the Swiss Tropical and Public Health Institute, Basel, provided data for Fig. 5, S1f, S2, and S3. FR is funded by the National Centres of Competence in Research (NCCRs) 'Molecular Systems Engineering'.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.softx.2021.100710>.

References

- [1] Kosack CS, Page A-L, Klatser PR. A guide to aid the selection of diagnostic tests. *Bull World Health Organ* 2017;95(9):639.
- [2] Koczula KM, Gallotta A. Lateral flow assays. *Essays Biochem* 2016;60(1):111–20.
- [3] O'Farrell B. Evolution in lateral flow-based immunoassay systems. In: *Lateral flow immunoassay*. Totowa, NJ: Humana Press; 2009, p. 1–33. http://dx.doi.org/10.1007/978-1-59745-240-3_1.
- [4] Pai NP, Vadnais C, Denking C, Engel N, Pai M. Point-of-care testing for infectious diseases: Diversity, complexity, and barriers in low- and middle-income countries. *PLoS Med* 2012;9(9). <http://dx.doi.org/10.1371/journal.pmed.1001306>.
- [5] World Health Organization, et al. Antigen-detection in the diagnosis of SARS-CoV-2 infection using rapid immunoassays: interim guidance, 11 September 2020. World Health Organization; 2020.
- [6] Centers for Disease Control and Prevention, et al. Interim guidelines for COVID-19 antibody testing. Atlanta, GA, USA: CDC; 2020.
- [7] Broich K. Antigen tests zum direkten Erregernachweis des coronavirus SARS-CoV-2. *BfArM* 2020. URL <https://antigentest.bfarm.de/ords/f?p=110:100:2696728796025:::~&tz=2:00>.
- [8] FDA. EUA Authorized Serology Test Performance. Center for Devices and Radiological Health, U.S. Food and Drug Administration; 2020. URL <http://www.fda.gov/medical-devices/coronavirus-disease-2019-covid-19-emergency-use-authorizations-medical-devices/eua-authorized-serology-test-performance>.
- [9] Armbruster DA, Pry T. Limit of blank, limit of detection and limit of quantitation. *Clin. Biochem. Rev* 2008;29(suppl. 1, August):S49–52.
- [10] Urusov AE, Zherdev AV, Dzantiev BB. Towards lateral flow quantitative assays: detection approaches. *Biosensors* 2019;9(3):89.
- [11] Minter C, Casanovas-Massana A, Li T, McDonald D, Wang L, Pan SH, et al. Multi-site validation of a SARS-cov-2 igg/igm rapid antibody detection kit. Cold Spring Harbor Laboratory Press; 2020, medRxiv.
- [12] Lee LG, Nordman ES, Johnson MD, Oldham MF. A low-cost, high-performance system for fluorescence lateral flow assays. *Biosensors* 2013;3(4):360–73.
- [13] Mak WC, Beni V, Turner AP. Lateral-flow technology: From visual to instrumental. *TRAC Trends Anal Chem* 2016;79:297–305.
- [14] Vashist SK, Mudanyali O, Schneider EM, Zengerle R, Ozcan A. Cellphone-based devices for bioanalytical sciences Multiplex Platforms in Diagnostics and Bioanalytics. *Anal Bioanal Chem* 2014;406(14):3263–77. <http://dx.doi.org/10.1007/s00216-013-7473-1>.
- [15] Lee S, Kim G, Moon J. Development of a Smartphone-based reading system for lateral flow immunoassay. *J Nanosci Nanotechnol* 2014;14(11):8453–7. <http://dx.doi.org/10.1166/jnn.2014.9920>.
- [16] Scherr TF, Gupta S, Wright DW, Haselton FR. Mobile phone imaging and cloud-based analysis for standardized malaria detection and reporting. *Sci Rep* 2016;6(June):1–9. <http://dx.doi.org/10.1038/srep28645>.
- [17] Dutta S. Point of care sensing and biosensing using ambient light sensor of smartphone: Critical review. *Trends Anal Chem* 2019;110:393–400. <http://dx.doi.org/10.1016/j.trac.2018.11.014>.
- [18] Lee C, Noh J, O'Neal SE, Gonzalez AE, Garcia HH, Handali S. Feasibility of a point-of-care test based on quantum dots with a mobile phone reader for detection of antibody responses. *PLoS Negl Trop Dis* 2019;13(10):1–11. <http://dx.doi.org/10.1371/journal.pntd.0007746>.
- [19] Huang S, Abe K, Bennett S, Liang T, Ladd PD, Yokobe L, et al. Disposable autonomous device for swab-to-result diagnosis of influenza. *Anal Chem* 2017;89(11):5776–83. <http://dx.doi.org/10.1021/acs.analchem.6b04801>.
- [20] Zhdanov A, Keefe J, Franco-Waite L, Konnaiyan KR, Pyatt A. Mobile phone based ELISA (MELISA). *Biosens Bioelectron* 2018;103(December 2017):138–42. <http://dx.doi.org/10.1016/j.bios.2017.12.033>.
- [21] Huber PJ, Ronchetti EM. Robust statistics. 2nd ed. 2009, p. 172. <http://dx.doi.org/10.1002/9780470434697>.
- [22] Rudolf F, Kaltenbach H-M, Linnik J, Ruf M-T, Niederhauser C, Nickel B, et al. Clinical characterisation of eleven lateral flow assays for detection of COVID-19 antibodies in a population. Cold Spring Harbor Laboratory Press; 2020, medRxiv.