pyPOCQuant

Andreas P. Cuny and Aaron Ponti

Jan 19, 2021

CONTENTS:

The tool pyPOCQuant aims to automatically detect and quantify signal bands from lateral flow assays (LFA) or Point of Care tests (POC or POCT) from an image. It can batch analyze large amounts of images in parallel. An analysis pipeline can be run either from the command line (good for automating large numbers of analysis) or from a desktop application.

CHAPTER

ONE

INSTALLATION

1.1 Stable release

1.1.1 As module

To install pyPOCQuantui, just run this command in your terminal:

```
$ pip install pyPOCQuant
```

Installing pyPOCQuantui this way ensures that you get always the latest release.

If you don't have pip installed, this Python installation guide can guide you through the process.

1.1.2 As stand alone executable

If you want to install pyPOCQuantui on your system without installing Python yourself just download the pre-compiled executable matching your operating system:

Install tesseract following these instructions depending your operating system:

Warning: Make sure tesseract is on PATH of your environment.

pypocquantui can then be used trough its graphical user interface (GUI) directly.

1.2 From sources

1.2.1 All platforms

The latest sources for pypocquantui can be downloaded from the Github repo.

pyPOCQuant requires python 3.6. It is recommended to use miniconda. When miniconda is installed, start the terminal and type:

1. Install system *Python3* or *miniconda3*.
2. Create a new environment "pyPOCQuantEnv" with:

```
$ conda create -n pyPOCQuantEnv python=3.6
$ activate pyPOCQuantEnv
```

Note: More information about conda environments can be found here

3. You can clone the public repository:

```
$ git clone git://git.gitlab.com/csb.ethz/pypocquantui.git
```

Once you have a copy of the source, navigate into the directory *pypocquantui* and install dependencies with:

Note: *${platform}* is one of *win32.txt*, *linux.txt*, or *osx.txt*.

Then to start the UI run:

```
$ fbs run
```

If you use PyCharm make sure you open the project with its root folder and add

> */pypocquantui/src/main/python/main.py*

to the run configuration.

1.2.2 Windows

- Install tesseract

1.2.3 Linux

Install the following dependences (instructions for Ubuntu Linux):

```
$ sudo apt install libzmq3-dev, tesseract-ocr, libzbar0
```

1.2.4 macOS

To install the required dependencies we recommend to use the packaging manager *brew*. Install it from here if you have't allready Install brew .

```
$ brew install zbar
$ brew install tesseract
```

1.3 Build from source

To compile and create a pyPOCQuantUI installer, perform following steps. In the following *{ppcqui_root}* points to the root folder of the *pyPOCQuantUI* checked-out code.

1.3.1 Windows

```
$ cd ${ppcqui_root}
$ python ./make_build.py
```

You will find the installer in *${ppcqui_root}targetpyPOCQuant*.

1.3.2 Linux

```
$ sudo apt install ruby ruby-dev rubygems build-essential
$ sudo gem install --no-document fpm
$ cd ${ppcqui_root}
$ python ./make_build.py
```

This will create a *${ppcqui_root}/target/pyPOCQuant/pyPOCQuant.deb* package that can be installed and redistributed.

```
$ sudo apt install ${ppcqui_root}/target/pyPOCQuant/pyPOCQuant.deb
```

Please notice that client machines will need to install also two dependences:

```
$ sudo apt install tesseract-ocr, libzbar0
$ sudo apt install ${ppcqui_root}/target/pyPOCQuant/pyPOCQuant.deb
```

1.3.3 macOS

```
$ cd ${ppcqui_root}
$ python ./make_build.py
```

Note:

- Depending on your Python installation, you may need to use *pip3* instead of *pip*.
- For both running it from source or with the compiled binaries *zbar* and *tesseract* needs to be installed and be on PATH. On Windows *zbar* libs are installed automatically.

CHAPTER

TWO

PYPOCQUANT USER MANUAL

2.1 Introduction

The tool **pyPOCQuant** aims to automatically detect and quantify signal bands from **lateral flow assays** (**LFA**) or **Point of Care tests** (**POC** or **POCT**) from an image. It can batch analyze large amounts of images in parallel.

An analysis pipeline can be run either from the command line (good for automating large numbers of analysis) or from a desktop application.

2.2 Command line workflow

1. Split images by POCT manufacturer if needed
2. Copy all the images of the same kind into one folder
3. Prepare a settings (configuration) file
4. Run the pipeline

2.2.1 Split images by POCT manufacturer

> This only applies if you collected many images using POCTs from different vendors and stored all the images in one common folder! Analysis settings would need to be slightly adapted for different POCTs shapes and sizes.

If you have many images in an unorganized way we provide a helper script to sort them by manufacturer into subfolders.

> This can also be run from the UI, see below.

```
$ python ./split_images_by_strip_type_parallel.py -f /PATH/TO/INPUT/FOLDER -o /PATH/
↪TO/OUTPUT/FOLDER -w ${NUM_WORKERS}
```

- `/PATH/TO/INPUT/FOLDER`: path to the folder that contains all images for a given camera.
- `PATH/TO/OUTPUT/FOLDER`: path where all images will be organized into subfolders; one per each strip manufactured. Strip images that cannot be recognized (or do not contain any strip) will be moved to an `UNDEFINED` subfolder.
  - Currently recognized manufacturers:
    * AUGURIX
    * BIOZAK

    * CTKBIOTECH
    * DRALBERMEXACARE
    * LUMIRATEK
    * NTBIO
    * SUREBIOTECH
    * TAMIRNA

Please notice: the list of known manufacturers is defined in `pyPOCQuant.consts.KnownManufacturers`.

- `NUM_WORKERS`: number of parallel processes; e.g. `8`.

2.2.2 Settings file preparation

You can prepare a default parameter file from the command line as follows:

```
$ python ./pypocquant/pyPOCQuant_FH.py -c /PATH/TO/INPUT/settings_file.conf
```

Open the file in a text editor and edit it.

```
qc=True
verbose=True
sensor_band_names=('igm', 'igg', 'ctl')
peak_expected_relative_location=(0.25, 0.53, 0.79)
sensor_center=(178, 667)
sensor_size=(61, 249)
sensor_border=(7, 7)
perform_sensor_search=True
qr_code_border=40
subtract_background=True
sensor_search_area=(71, 259)
sensor_thresh_factor=2.0
raw_auto_stretch=False
raw_auto_wb=False
strip_try_correct_orientation=False
strip_try_correct_orientation_rects=(0.52, 0.15, 0.09)
strip_text_to_search='COVID'
strip_text_on_right=True
force_fid_search=True
```

> Some of the parameter names contain the term `strip`: this is used to indicate the POCT. The prefix `sensor` indicates the measurement region within the `strip`.

See Explanations for detailed description of the parameters.

> Please notice that some parameters are considered "Advanced"; in the user interface the parameters are separated into "Runtime parameters", "Basic parameters", and "Advanced parameters".

How to determine the parameters manually

Open the settings file and adjust the parameters to fit your images.

Important parameters are the `sensor_size`, `sensor_center`, and `sensor_search_area` (the latter being an advanced parameter).

> The user interface allows to easily define those parameters by drawing onto the extracted POCT image.

Sensor parameters are relative to the POCT image.

In the following we show how to obtain position and extent of the sensor areas in Fiji or ImageJ. Later we will see how to do the same in the pyPOCQuant user interface.

- When drawing a rectangular region of interest, the size is displayed in Fiji's toolbar; e.g. `x=539, y=145, **w=230, h=62**`.
- When hovering over the central pixels in the top or left sides of the selection, the `x`, and `y` coordinates of the center, respectively, are show in Fiji's toolbar; e.g. `x=*601*, y=144, value=214` (and equivalently for `y`).

2.2.3 Run the pipeline

Run the analysis per manufacturer manually

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

- `/PATH/TO/INPUT/FOLDER/MANUFACTURER`: path to the folder that contains all images for a given camera and manufacturer.
- `/PATH/TO/RESULTS/FOLDER`: path where the results (and the quality control images) for a given camera and manufacturer will be saved. The results are saved in a `quantification_data.csv` text file.
- `/PATH/TO/CONFIG/FILE`: path to the configuration file to be used for this analysis. Please see below. Notice that a configuration file will be needed per manufacturer and (possibly) camera combination.
- `NUM_WORKERS`: number of parallel processes; e.g. `8`.

2.3 GUI workflow

1. Split images by POCT manufacturer if needed
2. Copy all the images of the same kind into one folder
3. Select the folder containing the images to be processed
4. Set all analysis parameters
5. Run the pipeline

2.3.1 Split images by POCT manufacturer

> This only applies if you collected many images using POCTs from different vendors and stored all the images in one common folder! Analysis settings would need to be slightly adapted for different POCTs shapes and sizes.

To do so go to `File` –> `Split images by type` to open the dialog to split the images.

How to determine the parameters automatically using the GUI

A settings file must not necessarily be created in advance. The Parameter Tree can be edited directly. Optionally, settings can be loaded or saved from the UI.

How to estimate sensor parameters graphically in the UI:

- `Select the input folder` and click on one of the listed images to display it. The POCT region will be automatically extracted and shown in the view at the top. Please mind that this can take a few seconds. The lower view shows the whole image.
- Hit the `Draw sensor outline` icon (red arrow) in the toolbar. This will allow you to interactively define the `sensor area` and the `peak_expected_relative_location` parameters.

Drawing sensor by clicking into the corners	Drawing finished with aligned bars

- Draw the four corners of the sensor and place the vertical bars on the bands. This will cause all relevant parameters to be populated in the Parameter Tree. Please notice that, by default, the `sensor_search_area` is set to be 10 pixels wider and taller than the `sensor_size`. This can be changed in the advanced parameters (but beware to keep it only slightly larger than the `sensor_size`: it is meant only for small refinements).
- You can test current parameters on one image by clicking the `Test parameters` button under the Parameter Tree.
- Optionally, you can save the settings file (Ctrl+S, `File->Save settings file`)

Run the analysis per manufacturer automatically using the GUI

Once the previous steps are done and all parameters are correctly set, you can hit the `Run` button to start the analysis.

*Note: a step by step guide can be found under **Quick start** (`Help -> Quick start`)*

2.4 Settings

The following settings must be specified. These are default values and need to be adopted for a series of the same kind of images. Please note: in the following, `strip` is used to indicate the POCT, and `sensor` to indicate the measurement region within the `strip`.

```
qc=True
verbose=True
sensor_band_names=('igm', 'igg', 'ctl')
peak_expected_relative_location=(0.25, 0.53, 0.79)
sensor_center=(178, 667)
sensor_size=(61, 249)
sensor_border=(7, 7)
```

(continues on next page)

(continued from previous page)

```
perform_sensor_search=True
qr_code_border=40
subtract_background=True
sensor_search_area=(71, 259)
sensor_thresh_factor=2.0
raw_auto_stretch=False
raw_auto_wb=False
strip_try_correct_orientation=False
strip_try_correct_orientation_rects=(0.52, 0.15, 0.09)
strip_text_to_search='COVID'
strip_text_on_right=True
force_fid_search=True
```

2.4.1 Explanations

Runtime parameters

max_workers

- The analysis can work in parallel. Specify the maximum number of images that are run in parallel. The maximum allowed value is the number of cores in your machine.

qc

- Toggle creation of quality control images.
- Possible values: `True` or `False`
- Recommended: `True` when testing parameters.

verbose

- Toggle extensive information logging.
- Possible values: `True` or `False`
- Recommended: `True` when testing parameters.

Basic parameters

number_of_sensor_bands

- It defines the number of test lines (TLs) to be expected in the POCT, including the control line. This parameter is used by the user interface to dynamically adapt the tree for related settings (see `sensor_band_names` and `peak_expected_relative_location` below), and is not part of the settings file, since it can be easily derived fro those parameters.
- Possible values: `2` to `100`

control_band_index

- Index of the control line.
- Possible values: `0`, `1`, `...`, `number_of_sensor_bands - 1`; or `-1` (last index).
- Default: `-1` (in Python parlance, `-1` means last index, or, the first index from the right).

sensor_band_names

- Custom name for the test lines (by default 3, needs to match the number of defined TLs `number_of_sensor_bands`) `t2`, `t1` and `ctl` (e.g., `IgM`, `IgG` and `Ctl`).

peak_expected_relative_location

- Expected relative peak positions as a function of the width of the sensor (= 1.0). These values can easily be set interactively using the UI.

sensor_center

- Coordinates in pixels of the center of the sensor with respect to the strip image: `(y, x)`.

sensor_size

- Area in pixels of the sensor to be extracted: `(height, width)`.

sensor_border

- Lateral and vertical sensor border in pixels to be ignored in the analysis to avoid border effects: `(lateral, vertical)`.

perform_sensor_search

- If `True`, the (inverted) sensor is searched within `sensor_search_area` around the expected `sensor_center`; if `False`, the sensor of size `sensor_size` is simply extracted from the strip image centered at the relative strip position `sensor_center`.
- Possible values: `True` or `False`
- Recommended: `True`

qr_code_border

- Lateral and vertical extension of the (white) border around each QR code.

subtract_background

- If `True`, estimate and subtract the background of the sensor intensity profile (bands).
- Possible values: `True` or `False`
- Recommended: `True`

Advanced parameters

These parameters will most likely work with the default values above.

sensor_search_area

- Search area in pixels around the sensor: `(height, width)`.
- Used only if `skip_sensor_search` is `False`.
- **Try to keep it just a bit larger than the sensor size: in particular, try to avoid picking up features (e.g. text) in close proximity of the sensor.**

sensor_thresh_factor

- Set the number of (robust) standard deviations away from the median band background for a peak (band) to be considered valid.
- Recommended: `2`, maybe `3`.

raw_auto_stretch

- Whether to automatically correct the white balance of RAW images on load. This does not affect JPEG images!
- Possible values: `True` or `False`
- Recommended: `False`

raw_auto_wb

- Whether to automatically stretch image intensities of RAW images on load. This does not affect JPEG images!
- Possible values: `True` or `False`
- Recommended: `False`

strip_try_correct_orientation

- Whether to automatically try to rotate a POCT that was mistakenly placed on the template facing the wrong direction (and where the control band is on the left instead of on the right). The pipetting inlet will be searched in the POCT; the inlet is assumed to be found on the side opposite to the control band, and always on the left. If found on the right, the image will be rotated.
- Possible values: `True` or `False`
- Default: `False`
- If set to `True`, make sure to properly set the `strip_try_correct_orientation_rects` parameters below!

strip_try_correct_orientation_rects

- Parameters for defining two rectangles left and right from the sensor center to be used to detect the pipetting inlet. The first parameter, `Relative height factor`, defines the relative height of the rectangles with respect to the strip. The second parameter, `Relative center cut-off`, defines the relative offset from the sensor center and therefore the width of the rectangle. Finally, the third parameter, `Relative border cut-off`, defines the relative offset from the strip's left and right borders and hence the width of the search rectangle.
- Possible values: `(0:1, 0:1, 0:1)`
- Default: `(0.52, 0.15, 0.09)`

strip_text_to_search

- Whether to use a specific text printed on the POCT to automatically try to rotate a POCT that was mistakenly placed on the template facing the wrong direction (and where the control band is on the left instead of on the right). Set to `""` to skip search and correction. If the strip has some text printed on either side of the sensor, it can be searched to guess the orientation. See also `strip_text_on_right`.

strip_text_on_right

- Assuming the strip is oriented horizontally, whether the `strip_text_to_search` text is expected to be on the right. If `strip_text_on_right` is `True` and the text is found on the left hand-side of the strip, the strip will be rotated 180 degrees.
- Ignored if `strip_text_to_search` is `""`.

force_fid_search

- If force fid search is activated, try hard (and slow!) to find an FID on a barcode or QR code label on the image identifying the sample.
- Possible values: `True` or `False`
- Recommended: `False`

2.5 Results

The analysis pipeline delivers a `.csv` that contains a relatively large table of results. The extracted features are explained in the following.

2.5.1 Result table

Structure and description of the result table:

`fid`: patient FID in the form `F5921788`

`fid_num`: just the numeric part of the FID (i.e., `5921788`)

`filename`: name of the analyzed image

`extension`: extension (either *.JPG or *.ARW, *.CR2, *.NEF)

`basename`: filename without extension

`iso_date`: date of image acquisition in the form YYYY-MM-DD (e.g. 2020-04-14)

`iso_time`: time of image acquisition in the form HH-MM-SS (24-h format)

`exp_time`: camera exposure time

`f_number`: aperture F number

`focal_length_35_mm`: 35mm equivalent focal length

`iso_speed`: camera ISO value

`manufacturer`: POCT manufacturer

`plate`: plate number

`well`: well (e.g. `A 01`)

`ctl`: 1 if the control band could be extracted, 0 otherwise.

`t2`: 1 if the `t2` band (e.g. IgM) could be extracted, 0 otherwise.

`t1`: 1 if the `t1` band (e.g. IgG) could be extracted, 0 otherwise.

`ctl_abs`: absolute signal strength of the control band,

`t2_abs`: absolute signal strength of the `t2` band,

`t1_abs`: absolute signal strength of the `t1` band.

`ctl_ratio`: relative signal strength of the control band (always 1.0 if detected)

`t2_ratio`: relative signal strength of the `t2` band with respect to the control band

`t1_ratio`: relative signal strength of the `t1` band with respect to the control band

`issue`: if issue is 0, the image could be analyzed successfully, if issue > 0 it could not. See the list of issues below

`user`: custom field

> Note: expect small residual variations in the absolute signal strengths (`ctl_abs`, `t2_abs`, and `t1_abs`) across images in a batch due to inhomogeneities in acquisition.
>
> Note 2: `ctl`, `t1`, and `t2` in the column names will be replaced by the names defines in `sensor_band_names`. For examples, `t1_ratio` may become `igg_ratio`.

Note 3: The number of test lines (TL) changes according to the parameter `number_of_sensor_bands`. By default, 3 TLs are defined including the `ctl` line. Changing the number of TLs also changes the number of columns in the results table.

Analysis issues

Each analyzed image is assigned an integer `issue`:

- 0: no issue, the analysis could be performed successfully
- 1: barcode extraction failed
- 2: FID extraction failed
- 3: strip box extraction failed
- 4: strip extraction failed
- 5: poor strip alignment
- 6: sensor extraction failed
- 7: peak/band quantification failed
- 8: control band missing

2.5.2 Quality control images

Types and examples' of quality control images:

Raw image shown as comparison:

- `IMAGE_FILE_NAME_aligned_box` Aligned raw image
- `IMAGE_FILE_NAME_box` : QR code box around the POCT oriented such that the control band is always on the right side.
- `IMAGE_FILE_NAME_rotated`: Raw image rotated such that the POCT is at the parallel to the bottom side of the image.
- `IMAGE_FILE_NAME_strip_gray_aligned`: Aligned POCT cropped around its outline such that it is parallel to the bottom side.
- `IMAGE_FILE_NAME_strip_gray_aligned_after_ocr` Aligned POCT cropped around its outline such that it is parallel to the bottom side after OCR filtering such that the pipetting part is always left (for the cases where the POCT was not placed in the correct orientation in the template.)
- `IMAGE_FILE_NAME_strip_gray_hough_analysis` Aligned POCT cropped around its outline such that it is parallel to the bottom side detecting the pipetting spot to identify wrongly oriented POCT in the strip box.
- `IMAGE_FILE_NAME_strip_gray_hough_analysis_candidates` Hough analysis candidate results. The rectangles indicate the search areas while as the circles indicate potential hits for the pipetting spot. Red rectangle and magenta circles identifies the side where the pipetting spot was detected. Note it is assumed that the control band is always opposite of the pipetting area.
- `IMAGE_FILE_NAME_sensor`: Aligned sensor crop showing the bands.
- `IMAGE_FILE_NAME_peak_overlays`: Sensor crop with colored rectangle overlay(s) indicating the area(s) where the signal for each detected band is quantified. Notice that the rectangle extends to cover the whole area under the curve, from background level through peak and back to background level.

- `IMAGE_FILE_NAME_peak_background_estimation`: Control figure displaying the performance of the background estimation fit. Black dashed line is a an estimation of the background level obtained by robust linear fit of the band profile. From the estimate background trend a constant value is subtracted (resulting red solid line). This is to make sure that the signal is flat after correction, but no values are clipped.
- `IMAGE_FILE_NAME_peak_analysis`: Control figure displaying the performance of the peak analysis. Red circle indicates the max peak height. The green dashed line is an estimate of the local background that is used to test all candidate local maxima against a threshold defined by the red dashed line. This line is calculated as the (median of the background values) + `f` * (median deviation of the background values). The factor `f` is a user parameter and defaults to 2. The solid blue, orange and green line under the curves indicate the local span of each of the bands and indicate which part of the signal is integrated.

2.5.3 Log file

The log file contains more detailed information for each processed image identified by its file name, such as `IMG_8489.JPG`.

It informs about barcode extraction and its rotation, QR code box rotation, FID extraction, actual sensor coordinates and the identified bands.

Example log:

```
File = IMG_8489.JPG
Processing IMG_8489.JPG
Best percentiles for barcode extraction: (0, 100); best scaling factor = 0.25; score␣
↪= 6/6
File IMG_8489.JPG: best percentiles for barcode extraction after rotation: (0, 100);␣
↪best scaling factor = 0.25; score = 6/6
File IMG_8489.JPG: Strip box image rotated by angle -0.9172186022623166 degrees using␣
↪QR code locations.
File IMG_8489.JPG: FID = 'F5923994'
File IMG_8489.JPG: sensor coordinates = [140, 207, 523, 780], score = 1.0
Peak 69 has lower bound 48 (d = 21) with relative intensity 0.06 and upper bound 93␣
↪(d = 24) with relative intensity 0.00. Band width is 46. Band skewness is 1.14
Peak 138 has lower bound 104 (d = 34) with relative intensity 0.00 and upper bound␣
↪162 (d = 24) with relative intensity 0.10. Band width is 59. Band skewness is 0.71
Peak 203 has lower bound 170 (d = 33) with relative intensity 0.04 and upper bound␣
↪248 (d = 45) with relative intensity 0.00. Band width is 79. Band skewness is 1.36
File IMG_8489.JPG: the bands were 'normal'.
✓ File IMG_8489.JPG: successfully processed and added to results table.
```

2.5.4 Settings file

A settings file is created in the `-o /PATH/TO/RESULTS/FOLDER` with the actually used parameters for the analysis. It can be used to reproduce the obtained results.

See settings file section for detailed description.

2.6 Graphical user interface

The GUI offers several actions via the menu, the toolbar and buttons.

1. `File menu`:
   - `File`: Lets you load (`File` –> `Load settings file`) and save (`File` –> `Save settings file`) a settings file
   - `Help`: Get quick instructions and open this manual
2. `Toolbar`:
   - `Load settings from file`: Load settings from file into the Parameter Tree.
   - `Save settings to file`: Save current settings to file.
   - `Draw sensor outline`: Activates drawing a polygon by clicking into the corners of the sensor on the images.
   - `Delete sensor`: Deletes currently drawn sensor.
   - `Mirror image vertically`: Mirrors the displayed image vertically.
   - `Mirror image horizontally`: Mirrors the displayed image horizontally.
   - `Rotate clockwise`: Rotates the displayed image clock wise.
   - `Rotate counter clockwise`: Rotates the displayed image counter clock wise.
   - `Set rotation angle in degrees`: Specifies the rotation angle.
   - `Zoom in`: Zooms in the displayed image.
   - `Zoom out`: Zooms out the displayed image .
   - `Reset zoom`: Resets the zoom level.
   - `Measure distance`: Lets you draw a line on the image to measure distances. It will update the `qr_border_distance` parameter.
   - `Show / hide console`: shows or hides the console at the bottom of the UI.
3. `Select input folder`: Allows to specify the input folder.

   `Select output folder`: (Optional) Lets you select a output folder. If left empty a output subfolder is automatically generated in the input folder.

   `Image list`: Lists all available images in the input folder. Click onto the filename to display one in **5**.
4. `Parameter Tree`: Adjust parameters manually if needed.
5. `POCT area`: Shows the extracted POCT and allows for drawing the sensor.
6. `Display area`: Shows the currently selected image.
7. `Test parameters`: Runs the pipeline on the selected image with current settings. The test folder will be opened automatically to inspect the control files.
8. `Run`: Runs the pipeline with the current settings**.
9. `Log`: Informs the user about performed actions.
10. `Tools menu`:
    - `Save POCT template`: Lets you save and print the POCT template to be used for the image acquisition.

- `Save QR labels template` Lets you save an Excel template to be used to generate QR code labels for all your samples from a list.
- *Generate QR labels*: Lets you generate QR labels for your samples using the excel template or a csv file with a list of the names in the correct format: `SAMPEID-MANUFACTURER-PLATE-WELL-USERDATA`. You can use the `USERDATA` field for **very short** annotations; please make sure not to use dashes (`-`) in this field, but replace them with underscores (_). You can define the page size, label size, position and number per page to match the format for any printable label paper as, for instance, from AVERY.
- `Split images by type`: helps organizing mixtures of images from different POCT manufacturers stored in one and the same folder. The tool will analyze each of the images and attempt to extract the manufacturer information from the QR code (if present) or by searching for the name on the POCT itself (OCR). If the manufacturer can be resolved, the image is moved into a sub-folder with the name of the manufacturer, otherwise it will be moved into a subfolder called UNDEFINED. As long as the QR code structure is satisfied (which is guaranteed if using the `Generate QR labels` tool explained above), even previously unknown manufacturers can be recognized. Should the QR code detection fail, a fall-back OCR detection can use a comma-separated list of expected manufacturer names to potentially identify unknown manufacturers. The `input folder` defines the folder containing all the images to process, and `output folder` the location where the split images should be moved. The `number of cores` defines the number of images that will be processed in parallel.

11. `Help menu`:
    - `Quick instructions`: Shows the quick instructions dialog.
    - `Quick start`: Opens the quick start document describing how to set up the image acquisition setup, perform the acquisition and some potential problems and their solutions one might encounter.
    - `User manual`: Opens this document.
    - `About`: About the software and its dependencies.

CHAPTER

THREE

USE AND EXAMPLES

The examples show the basic usage of **pyPOCQuant** to analyze images

3.1 pyPOCQuant quick start

For a reproducible and comparable analysis of your POCTs with **pyPOCQuant**, please carefully follow these instructions. They will show how to properly prepare the acquisition setup, the acquisition itself and the analysis of the images from lateral flow assays (LFA) / point of care tests (POCT).

This *quick start* guide focuses on the most relevant points. For detailed information read the relevant section in the *user manual* (Help -> Manual) .

3.1.1 Preparation of the imaging acquisition setup

Materials needed:

- Camera, for example SLR/mirror less (recommended, use raw and jpg), pocket camera, mobile phone
- POCT Template / mount.
- A tripod to mount the camera above the POCT template and mount. Alternatively, a box (plastic box, or even a shoe box) can be used to mount the camera at a defined distance above the POCT template.
- Tape, glue, scissors or scalpel to fix and build the mounts.
- Printer to print the POCT template and the sample QR labels.
- Power bar to charge the camera batteries or power it directly.
- Desktop computer or laptop to transfer the images and run **pyPOCQuant**.

Instruction to build the POCT mount with the POCT template

Print our generic template (get it from Help -> POCT template) in black and white (ideally on non glossy paper to avoid disturbing reflections) and place the POCT to evaluate in the center of the QR code box. Cut out its cartridge outline with a scalpel or scissors (Fig. 1). The fine red grid will help you to align the POCT nicely with the QR code box border. *Note: needs to be repeated for each cartridge design if its size changes.*

Glue or stick the template on one or two cartons and again cut out the region to place the POCT (Fig. 2). *Note: the narrower you cut the better it will hold the POCT at the exact same position.*

The basis of the template mount could also be 3D printed or laser cut from any material and aligned with the POCT template to build a solid POCT mount.

Fig. 1	Fig. 2

Instructions to build the photo box / acquisition station

While setting up the imaging acquisition station there are three important points to consider.

- First, make sure that you have **constant lightning conditions**. If just using the POCT template and a tripod (Fig. 3) make sure you have a dark room otherwise daylight changes will influence the images. Best would be using a photo box (Fig.4). *Note: Our POCT template changed over the course of the development but we don't have images of the setup from each stage. Here you see a very early incomplete version of it. Please us the one presented in Fig. 1.*
- Second, make sure that during a series of tests of the same kind **the camera is well fixed on the tripod**. Ideally you use the camera timer option or a remote control to release the images to make sure that the distance between the camera and the POCT on the POCT template is constant.
- Third, make sure that **the field of view does not change during a series**. For this the POCT template is well fixed on the table and the tripod with camera is not moved. If this is not the case, you will need to create a configuration file for each image, and will not be able to easily batch process them!

Fig. 3	Fig. 4

3.1.2 Image acquisition

Do **not write to or stick anything on the POCT**. Use the QR code labels instead to allow for machine readable identification of the sample and place the QR above the QR code box in its dedicated place.

Step 1:

- Check that **all QR codes on the template** are in the field of view and on the image.

Step 2:

- Check that the light conditions are constant and there are **no shadows on the POCT / sensor area** and there are **no reflections**.

Step 3:

- Check that there are no vibrations during the acquisition which could lead to a bad or blurred image. If possible, use **a remote control or computer control to take the images**. If not available, use the timer option carefully to avoid moving the camera.

Step 4:

- Check that the image **is sharp and in focus: in particular the POCT sensor area and the QR codes**.

Example image meeting all criteria's sufficiently except that the packaging is cut off. *Note: pyPOCQuant will detect the orientation of the image automatically. There is no need to rotate the images*

3.1.3 Analysis of the images with pyPOCQuant

- Follow the installer guide lines to install **pyPOCQuant**
- Install third-party dependences for your operating system.
  - Windows: *tesseract*: https://tesseract-ocr.github.io/tessdoc/Home.html.
  - Linux: `zbar` and `tesseract`.
    - In Ubuntu, you can install them with `sudo apt install libzbar0 tesseract-ocr`.
  - macOS: `cairo`, `zbar` and `tesseract`.
    - Using homebrew: `brew install cairo zbar tesseract`

Note: For most images it is sufficient to just load an image (Step 3 & 4) and draw the sensor (Step 5) and then test the automatically determined & default parameters with (Step 7) and finally run it on all images (Step 8).

Step 1:

- Copy the images of the same kind (i.e., same POCT cartridge / manufacturer and/or same imaging station, objective, distance to the sample) into a folder. *Note that the UI allows you to automatically split the images by manufacturer into subfolders (if included in the QR code labels); in addition, we provide a script to do so from the command line. For the details read the respective sections in the user manual (``Help -> User manual``).*

Step 2:

- Start **pyPOCQuant**.

Step 3:

- Select the image folder you want to analyze. Click on `Browse` input folder (`Ctrl+I`).
- (Optional) Click on `Browse` result folder to select the folder where to save results, logs and quality control images. By default, a subfolder `pipeline` is created in the input folder.

Step 4:

- Click on one image (ideally one which shows all bands) to load it. After a while (green progress bar fully to the right) the POCT area will be extracted and displayed on the top-right canvas.

Image selected - strip extraction pending	Image selected - strip extraction done and displayed

Step 5:

- Hit the draw sensor icon in the toolbar and click into the image to draw a rectangle around the sensor area.

  The parameters `sensor_center`, `sensor_size` and `sensor_search_area` will be set automatically in this step.

Click into the corners of the sensor to draw the sensor outline	Drawing finished. Parameters `sensor_center`, `sensor_size` and `sensor_search_area` have now been set automatically

Step 6:

- Adjust the expected position of the bands by clicking on the vertical violet lines and move them in place such that they are centered and overlapping with the bands on the test. Optionally, you can also fine-adjust by changing the parameters in the tree.

Not properly aligned control band line (vertical violet line). `peak_expected_relative_location`=(0.23, 0.5, 0.7)	Properly aligned band lines `peak_expected_relative_location`=(0.23, 0.5, 0.77)
	!

Step 7

- Change the band labels for t2, t1 and ctl band according to the test analyzed. For example `sensor_band_names=(IgG, IgM, Ctl)`. These names will be used as prefixes in the header of the result table.

Step 8:

- Hit `Test parameters` and check the result based on the quality control images. If you get false positive detections for weak signals increase the advanced parameter `sensor_thresh_factor` and hit test again.
- If the result looks good (check the quality control images `IMAGE_NAME_peak_overlays control image`, and `IMAGE_NAME_peak_analysis control image` and the entries in the `quantification data.csv` file), you can continue. Otherwise adjust the parameters further, look up the advanced parameters in the manual, or check the common problems and solutions below.

IMAGE_NAME_peak_overlays control image	IMAGE_NAME_peak_analysis control image

Step 9:

- Hit `Run` to batch analyze all images in the folder in parallel.

Repeat the procedure for all other folders. *Note: if the POCT cartridge design changes or a different camera with a different perspective is used, a new configuration file has to be generated and tested. Otherwise, one can load the same configuration file also for other / new images. To load a configuration file just double-click on it if it is in the same folder as the input images, or hit `Ctrl+O` or select `File -> Load settings from file*`

3.1.4 Potential problems and their solution:

Problem: There are artifacts / weak signals that get quantified wrongly as a band (Fig. 5)

Solution: Increase the `sensor threshold factor` (Fig. 6)

Fig 5 `sensor threshold factor=1`	Fig 6 `sensor threshold factor=2`

Problem: *One or more bands were missed* or *the wrong band(s) were extracted* (Fig. 7)

Solution: Adjust the `Peak expected relative location` parameter for the band(s) which were not detected. If that did not solve the problem check the quality-control images if the sensor was detected correctly. If not adjust the sensor position and its size (Fig. 8).

Fig 7	Fig 8

Problem: Almost no pixels are considered for quantification (Fig. 9)

Solution: Reduce the `sensor border x|y` values to consider more pixels of the sensor (Fig. 10). If it considers too many pixels increase the parameter values.

Problem: I have a lot of images to be processed and it is slow.

Solution: Increase the `Number of cores` parameter to the maximum of your computer. Use a more powerful station or cluster.

Problem: By accident, the image was taken with the POCT wrongly oriented and the control band is left (Fig. 11).

Solution: Select the checkbox `try to correct strip orientation`. This will try to rotate the image correctly for the analysis (Fig 12). The qc image lets you verify if the correction works. If it does not work modify the parameters (*Relative height factor*, *Relative center cut-off*, *Relative border cut-of*) defining the size and position of the search rectangles. The red rectangle (Fig. 11) indicates where the inlet was found and will rotate the image such that the inlet is left and the control band on the right (Fig. 12). The search rectangles should only include the region around the pipetting inlet. If it still does not work, the last chance is to try and search for some text printed on one side of the POCT. Add the prominent text (for example, "COVID" as in Figg. 11 or 12) to the `Strip text to search (orientation)` parameter and select if the text is on the right or not (check or uncheck the `Strip text is on the right` parameter). If this still fails, the image will have to be discarded and a new one will need to be reacquired.

Fig 11	Fig 12

Problem: I have a lot of images from with different POCT cartridge designs from different manufacturers taken with the same camera but my configuration file does only work for one type.

Solution: Split the images into a subfolder for each cartridge design / manufacturer. If you used the QR code sample labels you can use the script described in the manual to do this automatically for you.

Problem: I have a lot of images with different POCT cartridge designs from different manufacturers. Do I really need a separate configuration for each design?

Solution: Unfortunately yes. As they come in any shape the software needs some specific guidance to know where to search for the bands and to allow for robust and reproducible results. One solution to relax this assumption would be to change the POCT cartridge design by including small qr codes directly next to the sensor. That would allow us also to get rid of the QR code template. If you have a direct contact to your favorite manufacturer, tell them about it and their potential competitive advantage in the market (Fig 13)!

Fig 13

3.2 Command line usage

pyPOCQuant can be used trough its command line interface (cli). It is convenient to process a large amount of different folder trough i.e a bash script.

To show the usage type:

```
python -m pypocquant.pyPOCQuant --help
```

To run the pipline for a given folder and config type:

```
python -m pypocquant.pyPOCQuant_FH -f path/to/images -s path/to/config.conf -w 10
```

To split and organize images of different kinds in one folder type:

```
python -m pypocquant.split_images_by_strip_type_parallel -f path/to/images -o path/to/
↪result_dir -w 10
```

3.3 Scripting

pyPOCQuant can be used directly from within python scripts and therefore being part of a larger workflow. It is convenient to process a large amount of different folder automatically. Or further automatically process results and generation of reports.

Minimal example with default settings. Add the following code to a file such as *example.py* wile replaceing the *input_folder_path* and *results_folder_path* to the example or your images :

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

# Get the default settings
settings = default_settings()

# Change settings manually as needed
settings["sensor_band_names"] = ('igm', 'igg', 'ctl')

# Alternatively, load existing settings file
# from pypocquant.lib.settings import load_settings
# settings = load_settings('full/path/to/settings/file.conf')

# Set final argument
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
)
```

and run it with:

```
python -m example.py
```

3.4 pyPOCQuant with Jupyter

Demo notebook to show the usage of **pyPOCQuant** using a Jupyter/IPython notebook.

This is a convinient way to automate the execution of multiple folders and directly analize and plot the results.

[4]:
```
# Load the relevant dependencies
import os
import sys
```

(continues on next page)

(continued from previous page)

```
import pandas as pd
sys.path.append('..\..')
from pypocquant.lib.pipeline import run_pipeline
from pypocquant.lib.settings import load_settings
from pathlib import Path
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from IPython.display import Image, display
```

Next lets load and print the example config

```
[8]: p = Path(os.path.abspath('images'))
settings_file_path = Path(p.parent / 'config.conf')
print('The pipeline will be run with the following configuration')
f = open(str(settings_file_path), "r")
print(f.read())
```

```
The pipeline will be run with the following configuration
max_workers=4
qc=True
verbose=True
control_band_index=-1
sensor_band_names=('igm', 'igg', 'ctl')
peak_expected_relative_location=(0.31, 0.53, 0.75)
sensor_center=(147, 522)
sensor_size=(45, 215)
sensor_border=(7, 7)
perform_sensor_search=True
qr_code_border=40
subtract_background=True
sensor_search_area=(55, 225)
sensor_thresh_factor=2.0
raw_auto_stretch=False
raw_auto_wb=False
strip_try_correct_orientation=False
strip_try_correct_orientation_rects=(0.52, 0.15, 0.09)
strip_text_to_search=''
strip_text_on_right=False
force_fid_search=False
```

```
[9]: if settings_file_path.exists():
    input_folder_path = Path(p)
    results_folder_path = Path(input_folder_path / 'pipeline')
    results_folder_path.mkdir(parents=True, exist_ok=True)
    print(f'RUN pipeline for {p}')

    ## Load the settings
    settings = load_settings(settings_file_path)

    ## Run the pipeline
    run_pipeline(
        input_folder_path,
        results_folder_path,
        **settings
    )
```

```
  0%|                                                                          ␣
↪           | 0/7 [00:00<?, ?it/s]
```

```
RUN pipeline for C:\Users\Localadmin\Documents\pypocquantui\src\main\python\
↪pyPOCQuantUI\pypocquant\examples\images
```

```
100%|| 7/7 [00:10<00:00,  1.52s/it]
```

```
Results written to C:\Users\Localadmin\Documents\pypocquantui\src\main\python\
↪pyPOCQuantUI\pypocquant\examples\images\pipeline\quantification_data.csv
Logfile written to C:\Users\Localadmin\Documents\pypocquantui\src\main\python\
↪pyPOCQuantUI\pypocquant\examples\images\pipeline\log.txt
Settings written to C:\Users\Localadmin\Documents\pypocquantui\src\main\python\
↪pyPOCQuantUI\pypocquant\examples\images\pipeline\settings.conf
Pipeline completed.
```

3.4.1 Read and plot the results

[10]:
```
results = pd.read_csv(str(Path(p / 'pipeline/quantification_data.csv')))
results
```

[10]:
```
                fid         fid_num      filename extension  basename  \
0  H01601828610122  1601828610122  IMG_9067.JPG      .JPG  IMG_9067
1         F5921394        5921394  IMG_9068.JPG      .JPG  IMG_9068
2         F5922180        5922180  IMG_9069.JPG      .JPG  IMG_9069
3         F5922944        5922944  IMG_9070.JPG      .JPG  IMG_9070

     iso_date  iso_time exp_time f_number  focal_length_35_mm  ... igm  \
0  2020-06-21  12-14-03   [1/10]  [63/10]                  -1  ...   0
1  2020-06-21  12-14-46   [1/10]  [63/10]                  -1  ...   1
2  2020-06-21  12-15-07   [1/10]  [63/10]                  -1  ...   1
3  2020-06-21  12-15-28   [1/10]  [63/10]                  -1  ...   0

      igm_abs  igm_ratio igg      igg_abs  igg_ratio  ctl      ctl_abs  \
0    0.000000   0.000000   0     0.000000   0.000000    1  1087.000925
1  822.735949   0.805935   1  1443.824091   1.414340    1  1020.846550
2  200.968865   0.194804   1  1029.903783   0.998311    1  1031.645989
3    0.000000   0.000000   1   405.080753   0.403488    1  1003.946928

   ctl_ratio   user
0        1.0  CUNYA
1        1.0  CUNYA
2        1.0  CUNYA
3        1.0  CUNYA

[4 rows x 25 columns]
```

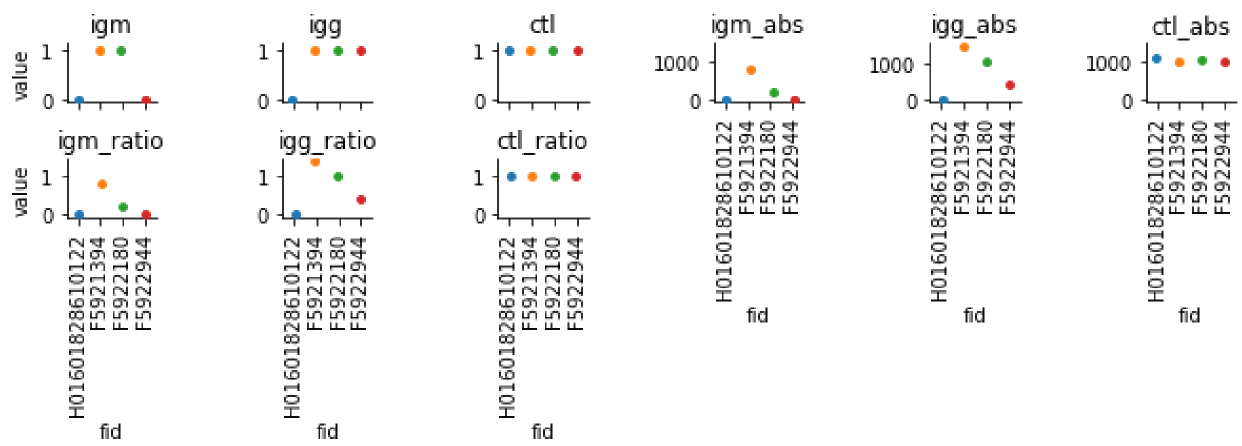
[11]:
```
%matplotlib inline
dm = pd.melt(results, id_vars=['fid'], value_vars=['igm', 'igg', 'ctl', 'igm_abs',
↪'igg_abs', 'ctl_abs', 'igm_ratio', 'igg_ratio', 'ctl_ratio'])
g = sns.catplot(x="fid", y="value", hue="fid", col='variable', col_wrap=6, data=dm,␣
↪sharey=False, height=1, aspect=1.6)
g.set_xticklabels(rotation=90)

axes = g.axes.flatten()
```

(continues on next page)

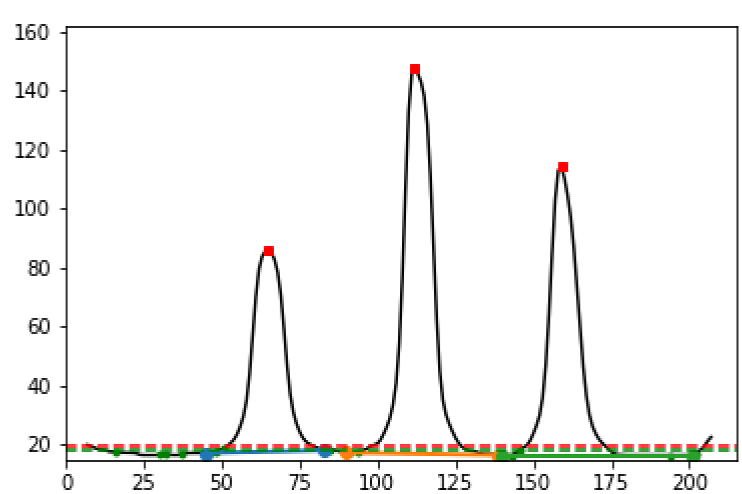
(continued from previous page)

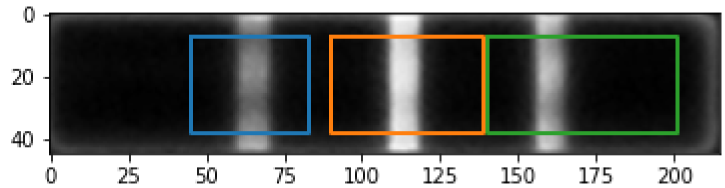
```
titles = ['igm', 'igg', 'ctl', 'igm_abs', 'igg_abs', 'ctl_abs', 'igm_ratio', 'igg_
↪ratio', 'ctl_ratio']
limits = [[-0.05,1.15], [-0.05,1.15], [-0.05,1.15], [-80,1500], [-80,1550], [-80,
↪1500],
          [-0.08,1.5], [-0.08,1.5], [-0.08,1.5]]
for ix, ax in enumerate(axes):
    ax.set_title(titles[ix])
    ax.set_ylim(limits[ix])
g.savefig("output.pdf", dpi=300)
```

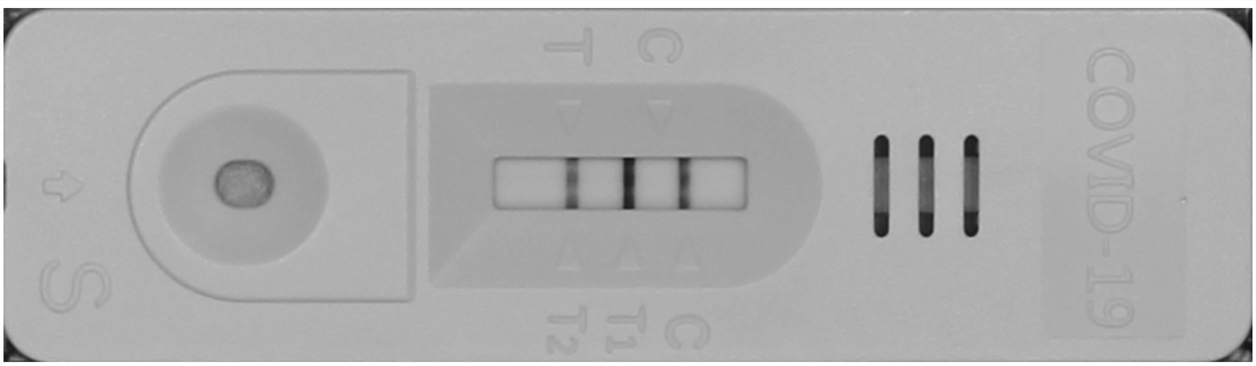


3.4.2 Check the qc images

[12]:
```
i1 = Image(filename=str(Path(p / 'pipeline/IMG_9068_JPG_peak_analysis.PNG')))
i2 = Image(filename=str(Path(p / 'pipeline/IMG_9068_JPG_peak_overlays.PNG')))
i3 = Image(filename=str(Path(p / 'pipeline/IMG_9068_JPG_strip_gray_aligned.PNG')))
i4 = Image(filename=str(Path(p / 'pipeline/IMG_9068_JPG_rotated.JPG')))
display(i1, i2, i3, i4)
```




0
20
40
0
25
50
75
100
125
150
175
200


T
C
S
T2
T1
C
COVID-19

3.4.3 Inspect the log file

[14]:
```
f = open(str(Path(p / 'pipeline/log.txt')), "r")
print(f.read())
f.close()
```

```
File = 20210105_config_run_1.conf

File = IMG_9067.JPG
Processing IMG_9067.JPG
Best percentiles for barcode extraction: (0, 100); best scaling factor = 0.25; score␣
↪= 6/6
Detected FIDs for rotated image: H01601828610122 H01601828610122
File IMG_9067.JPG: best percentiles for barcode extraction after rotation: (0, 100);␣
↪best scaling factor = 0.5; score = 6/6
File IMG_9067.JPG: Strip box image rotated by angle -0.285863954857813 degrees using␣
↪QR code locations.
File IMG_9067.JPG: FID = 'H01601828610122'
File IMG_9067.JPG: sensor coordinates = [126, 171, 423, 638], score = 1.0
Peak 143 has lower bound 99 (d = 44) with relative intensity 0.00 and upper bound 166␣
↪(d = 23) with relative intensity 0.00. Band width is 68. Band skewness is 0.52
File IMG_9067.JPG: the bands were 'normal'.
^aœ" File IMG_9067.JPG: successfully processed and added to results table.

File = IMG_9068.JPG
```

(continues on next page)

(continued from previous page)

```
Processing IMG_9068.JPG
Best percentiles for barcode extraction: (0, 100); best scaling factor = 0.25; score␣
↪= 6/6
Detected FIDs for rotated image: F5921394 F5921394
File IMG_9068.JPG: best percentiles for barcode extraction after rotation: (0, 100);␣
↪best scaling factor = 0.25; score = 6/6
File IMG_9068.JPG: Strip box image rotated by angle -0.32740089084438817 degrees␣
↪using QR code locations.
File IMG_9068.JPG: FID = 'F5921394'
File IMG_9068.JPG: sensor coordinates = [126, 171, 416, 631], score = 1.0
Peak 65 has lower bound 45 (d = 20) with relative intensity 0.00 and upper bound 83␣
↪(d = 18) with relative intensity 0.01. Band width is 39. Band skewness is 0.90
Peak 112 has lower bound 90 (d = 22) with relative intensity 0.01 and upper bound 142␣
↪(d = 30) with relative intensity 0.00. Band width is 53. Band skewness is 1.36
Peak 159 has lower bound 138 (d = 21) with relative intensity 0.00 and upper bound␣
↪201 (d = 42) with relative intensity 0.00. Band width is 64. Band skewness is 2.00
File IMG_9068.JPG: the bands were 'normal'.
^aœ" File IMG_9068.JPG: successfully processed and added to results table.

File = IMG_9069.JPG
Processing IMG_9069.JPG
Best percentiles for barcode extraction: (0, 100); best scaling factor = 0.5; score =␣
↪6/6
Detected FIDs for rotated image: F5922180 F5922180
File IMG_9069.JPG: best percentiles for barcode extraction after rotation: (0, 100);␣
↪best scaling factor = 0.5; score = 6/6
File IMG_9069.JPG: Strip box image rotated by angle -0.28627203365974213 degrees␣
↪using QR code locations.
File IMG_9069.JPG: FID = 'F5922180'
File IMG_9069.JPG: sensor coordinates = [126, 171, 416, 631], score = 1.0
Peak 65 has lower bound 43 (d = 22) with relative intensity 0.00 and upper bound 79␣
↪(d = 14) with relative intensity 0.03. Band width is 37. Band skewness is 0.64
Peak 112 has lower bound 89 (d = 23) with relative intensity 0.00 and upper bound 143␣
↪(d = 31) with relative intensity 0.00. Band width is 55. Band skewness is 1.35
Peak 158 has lower bound 133 (d = 25) with relative intensity 0.00 and upper bound␣
↪200 (d = 42) with relative intensity 0.00. Band width is 68. Band skewness is 1.68
File IMG_9069.JPG: the bands were 'normal'.
^aœ" File IMG_9069.JPG: successfully processed and added to results table.

File = IMG_9070.JPG
Processing IMG_9070.JPG
Best percentiles for barcode extraction: (0, 100); best scaling factor = 0.25; score␣
↪= 6/6
Detected FIDs for rotated image: F5922944 F5922944
File IMG_9070.JPG: best percentiles for barcode extraction after rotation: (0, 100);␣
↪best scaling factor = 0.5; score = 6/6
File IMG_9070.JPG: Strip box image rotated by angle -0.4086648209901469 degrees using␣
↪QR code locations.
File IMG_9070.JPG: FID = 'F5922944'
File IMG_9070.JPG: sensor coordinates = [126, 171, 424, 639], score = 1.0
Peak 96 has lower bound 76 (d = 20) with relative intensity 0.01 and upper bound 122␣
↪(d = 26) with relative intensity 0.00. Band width is 47. Band skewness is 1.30
Peak 141 has lower bound 122 (d = 19) with relative intensity 0.01 and upper bound␣
↪190 (d = 49) with relative intensity 0.00. Band width is 69. Band skewness is 2.58
File IMG_9070.JPG: the bands were 'normal'.
^aœ" File IMG_9070.JPG: successfully processed and added to results table.
```

(continues on next page)

(continued from previous page)

```
File = pipeline

File = test
```

```
c:\users\localadmin\appdata\local\programs\python\python36\lib\site-packages\
↪ipykernel_launcher.py:1: ResourceWarning: unclosed file <_io.TextIOWrapper name='C:\
↪\Users\\Localadmin\\Documents\\pypocquantui\\src\\main\\python\\pyPOCQuantUI\\
↪pypocquant\\examples\\images\\pipeline\\log.txt' mode='r' encoding='cp1252'>
  """Entry point for launching an IPython kernel.
```

CHAPTER

FOUR

LICENSE

pyPOCQuant is released under the **GPL v3** license:

4.1 GNU General Public License

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. http://fsf.org/

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

4.1.1 Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: **(1)** assert copyright on the software, and **(2)** offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

4.1.2 TERMS AND CONDITIONS

0. Definitions

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that **(1)** displays an appropriate copyright notice, and **(2)** tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that **(a)** is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and **(b)** serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users’ Legal Rights From Anti-Circumvention Law

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work’s users, your or third parties’ legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies

You may convey verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- **a)** The work must carry prominent notices stating that you modified it, and giving a relevant date.
- **b)** The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- **c)** You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- **d)** If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- **a)** Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- **b)** Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either **(1)** a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or **(2)** access to copy the Corresponding Source from a network server at no charge.
- **c)** Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- **d)** Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- **e)** Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either **(1)** a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or **(2)** anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- **a)** Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- **b)** Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- **c)** Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- **d)** Limiting the use for publicity purposes of names of licensors or authors of the material; or

- **e)** Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- **f)** Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated **(a)** provisionally, unless and until the copyright holder explicitly and finally terminates your license, and **(b)** permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either **(1)** cause the Corresponding Source to be so available, or **(2)** arrange to deprive yourself of the benefit of the patent license for this particular work, or **(3)** arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license **(a)** in connection with copies of the covered work conveyed by you (or copies made from those copies), or **(b)** primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

4.1.3 How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, your program’s commands might be different; for a GUI interface, you would use an “about box”.

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see http://www.gnu.org/licenses/.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read http://www.gnu.org/philosophy/why-not-lgpl.html.

CHAPTER

FIVE

AUTHORS

- Andreas P. Cuny <andreas.cuny at bsse dot ethz dot ch>
- Aaron Ponti <aaron.ponti at bsse dot ethz dot ch>

CHAPTER

SIX

CITING PYPOCQUANT

If you find pyPOCQuant useful please cite our paper:

> Cuny, A. P., Rudolf, F., & Ponti, A. (2020). pyPOCQuant - A tool to automatically quantify Point-Of-Care Tests from images. MedRxiv,. https://doi.org/10.1101/2020.11.08.20227470

or this repository using its DOI as follows:

> https://doi.org/10.1101/ (update once available)

Note: this DOI will resolve to the first version of pyPOCQuant.

```
@article{cuny2020,
        author    = {Andreas P. Cuny and Fabian Rudolf and Aaron Ponti},
        title     = {A tool to automatically quantify Point-Of-Care Tests from images}
↪,
        journal   = {MedRxiv},
        year      = {2020},
        doi       = {10.1101/2020.11.08.20227470}
        }
```

CHAPTER

SEVEN

API REFERENCE

This page contains auto-generated API reference documentation[1].

7.1 pypocquant

7.1.1 Subpackages

pypocquant.lib

Submodules

pypocquant.lib.analysis

Module Contents

Functions

get_min_dist(xy1, xy2)	Determine the minimal euclidean distance of a set of coordinates.
identify_bars_alt(peak_positions: list, profile_length: int, sensor_band_names: Tuple[str, . . .], expected_relative_peak_positions: Tuple[float, . . .], tolerance: float = 0.1)	Assign the peaks to the corresponding bar based on the known relative position in the sensor.
invert_image(image, bit_depth=8)	Inverts an image.
local_minima(array, min_distance=1)	Find all local minima of the array, separated by at least min_distance.
_find_lower_background(profile: np.ndarray, peak_index: int, lowest_bound: int, max_skip: int = 1)	This method is used by find_peak_bounds() and is not meant to be used as
_find_upper_background(profile: np.ndarray, peak_index: int, highest_bound: int, max_skip: int = 1)	This method is used by find_peak_bounds() and is not meant to be used as
find_peak_bounds(profile, border, peak_index, image_log, verbose=False)	Find the lower and upper bounds of current band.
fit_and_subtract_background(profile, border, subtract_offset=10)	Use a robust linear estimator to estimate the background of the profile and subtract it.

continues on next page

[1] Created with sphinx-autoapi

Table 1 – continued from previous page

estimate_threshold_for_significant_peaks(profile: np.ndarray, border_x: int, thresh_factor: float)	Estimate threshold for significant peaks in sensor signal.
analyze_measurement_window(window: np.ndarray, border_x: int = 10, border_y: int = 5, thresh_factor: float = 3.0, peak_width: int = 7, sensor_band_names: Tuple[str, ...] = ('igm', 'igg', 'ctl'), peak_expected_relative_location: Tuple[float, ...] = (0.27, 0.55, 0.79), control_band_index: int = -1, subtract_background: bool = False, qc: bool = False, verbose: bool = False, out_qc_folder: Union[str, Path] = '', basename: str = '', image_log: list = [])	Quantify the band signal across the sensor.
extract_inverted_sensor(gray, sensor_center=(119, 471), sensor_size=(40, 190))	Returns the sensor area at the requested position without searching.
get_sensor_contour_fh(strip_gray, sensor_center, sensor_size, sensor_search_area, peak_expected_relative_location, control_band_index=-1, min_control_bar_width=7)	Extract the sensor area from the gray strip image.
extract_rotated_strip_from_box(box_gray, box)	Segments the strip from the box image and rotates it so that it is horizontal.
adapt_bounding_box(bw, x0, y0, width, height, fraction=0.75)	Make the bounding box come closer to the strip by remove bumps along the outline.
point_in_rect(point, rect)	Check if the given point (x, y) is contained in the rect (x0, y0, width, height).
get_rectangles_from_image_and_rectangle_props(img_shape, rectangle_props=(0.52, 0.15, 0.09))	Calculate the left and right rectangles to be used for the orientation
use_hough_transform_to_rotate_strip_if_needed(img_gray, rectangle_props=(0.52, 0.15, 0.09), stretch=False, img=None, qc=False)	Estimate the orientation of the strip looking at features in the area around the
use_ocr_to_rotate_strip_if_needed(img_gray, img=None, text='COVID', on_right=True)	Try reading the given text on the strip. The text is expected to be on one
read_patient_data_by_ocr(image, known_manufacturers=consts.KnownManufacturers)	Try to extract the patient data by OCR.

pypocquant.lib.analysis.**get_min_dist**(*xy1*, *xy2*)

Determine the minimal euclidean distance of a set of coordinates.

Parameters

- **xy1** – First set of coordinates
- **xy2** – Second set of ccordinates

Returns Minimal distance

Return type tuple

pypocquant.lib.analysis.**identify_bars_alt**(*peak_positions: list*, *profile_length: int*, *sensor_band_names: Tuple[str, ...]*, *expected_relative_peak_positions: Tuple[float, ...]*, *tolerance: float = 0.1*)

Assign the peaks to the corresponding bar based on the known relative position in the sensor.

Parameters

- **peak_positions** – list List of absolute peak positions in pixels.
- **profile_length** – Length of the profile in pixels.

- **sensor_band_names** – Tuple[str, …] Tuple of sensor band names.
- **expected_relative_peak_positions** – Tuple[float, … } Tuple of expected relative (0.0 -> 1.0) peak positions.
- **tolerance** – Distance tolerance between pean position and expected position for assignment.

Returns dictionary of band assignments: {bar_name: index}

pypocquant.lib.analysis.**invert_image**(*image*, *bit_depth=8*)

Inverts an image.

Parameters

- **image** – Image to be inverted
- **bit_depth** – Bit depth of image

Returns image_inv: Inverted image.

Return type uint8

pypocquant.lib.analysis.**local_minima**(*array*, *min_distance=1*)

Find all local minima of the array, separated by at least min_distance.

Parameters

- **array** – Signal array
- **min_distance** – Minimal distance for local minima seperation

Returns array: Array with local minimas

Return type np.array

pypocquant.lib.analysis.**_find_lower_background**(*profile: np.ndarray*, *peak_index: int*, *lowest_bound: int*, *max_skip: int = 1*)

This method is used by find_peak_bounds() and is not meant to be used as a standalone method.

Parameters

- **profile** (*np.ndarray*) – Signal profile
- **peak_index** (*int*) – Index of the peak
- **lowest_bound** (*int*) – Highest bound
- **max_skip** (*int*) – Max skip

Returns current_lower_bound: Upper bound

Returns current_lower_background: Upper background

Returns d_lower:

pypocquant.lib.analysis.**_find_upper_background**(*profile: np.ndarray*, *peak_index: int*, *highest_bound: int*, *max_skip: int = 1*)

This method is used by find_peak_bounds() and is not meant to be used as a standalone method.

Parameters

- **profile** (*np.ndarray*) – Signal profile
- **peak_index** (*int*) – Index of the peak
- **highest_bound** (*int*) – Highest bound
- **max_skip** (*int*) – Max skip

Returns current_upper_bound: Upper bound

Returns current_upper_background: Upper background

Returns d_upper:

pypocquant.lib.analysis.**find_peak_bounds**(*profile*, *border*, *peak_index*, *image_log*, *verbose=False*)

Find the lower and upper bounds of current band.

Parameters

- **profile** (*np.ndarray*) – Signal profile
- **border** (*int*) – Border offset
- **peak_index** (*int*) – Index of the peak
- **image_log** (*list*) – Image log

Returns current_lower_bound: Lower bound

Returns current_upper_bound: Upper bound

Returns image_log: Log for this image

pypocquant.lib.analysis.**fit_and_subtract_background**(*profile*, *border*, *subtract_offset=10*)

Use a robust linear estimator to estimate the background of the profile and subtract it.

Parameters

- **profile** (*np.ndarray*) – Signal profile
- **border** (*int*) – Border offset
- **subtract_offset** (*int*) – Fixed offset to be used for substraction.

Returns profile: Background corrected profile.

Returns background: Estimated background.

Returns background_offset: Background offset.

pypocquant.lib.analysis.**estimate_threshold_for_significant_peaks**(*profile: np.ndarray*, *border_x: int*, *thresh_factor: float*)

Estimate threshold for significant peaks in sensor signal.

Parameters

- **profile** (*np.ndarray*) – Signal profile
- **border_x** (*int*) – Border offset in x
- **thresh_factor** (*float*) – Treshold factor for estimation.

Returns peak_threshold:

Returns loc_min_indices

Returns md

Returns lowest_background_threshold

`pypocquant.lib.analysis.`**`analyze_measurement_window`**(*window: np.ndarray*, *border_x: int = 10*, *border_y: int = 5*, *thresh_factor: float = 3.0*, *peak_width: int = 7*, *sensor_band_names: Tuple[str, ...] = ('igm', 'igg', 'ctl')*, *peak_expected_relative_location: Tuple[float, ...] = (0.27, 0.55, 0.79)*, *control_band_index: int = - 1*, *subtract_background: bool = False*, *qc: bool = False*, *verbose: bool = False*, *out_qc_folder: Union[str, Path] = ''*, *basename: str = ''*, *image_log: list = []*)

Quantify the band signal across the sensor.

Notice: the expected relative peak positions for the original strips were: [0.30, 0.52, 0.74]

Parameters

- **`window`** (*`np.ndarray`*) – Window (image) to be analyzed.
- **`border_x`** (*`int`*) – Border offset in x from window.
- **`border_y`** (*`int`*) – Border offset in y from window.
- **`thresh_factor`** (*`float`*) – Threshold factor from background.
- **`peak_width`** (*`int`*) – Minimal width of a peak.
- **`sensor_band_names`** (*`[str, ..]`*) – Names of the sensor bands (test lines TL).
- **`peak_expected_relative_location`** (*`tuple[float, ..]`*) – Tuple of relative expected peak positions in respect to the window.
- **`control_band_index`** (*`int`*) – Index of the control band for the list *sensor_band_names*.
- **`subtract_background`** (*`bool`*) – Bool to substract background.
- **`qc`** (*`bool`*) – Bool to retrun qc image.
- **`verbose`** (*`bool`*) – Bool to return verbose logging information
- **`out_qc_folder`** (*`Path`*) – QC image output folder
- **`basename`** (*`str`*) – Basename
- **`image_log`** (*`list`*) – Image log list.

Returns merged_results: Merged results

Returns image_log Image log

`pypocquant.lib.analysis.`**`extract_inverted_sensor`**(*gray*, *sensor_center=(119, 471)*, *sensor_size=(40, 190)*)

Returns the sensor area at the requested position without searching.

Parameters

- **`gray`** – Gray image.
- **`sensor_center`** – Sensor center coordinate on gray image.
- **`sensor_size`** – Sensor size on gray image.

Returns inverted_image Returns the extracted sensor on an inverted image.

`pypocquant.lib.analysis.`**`get_sensor_contour_fh`**(*strip_gray*, *sensor_center*, *sensor_size*, *sensor_search_area*, *peak_expected_relative_location*, *control_band_index=-1*, *min_control_bar_width=7*)

Extract the sensor area from the gray strip image.

Parameters

- **`strip_gray`** – np.ndarray Gray-value image of the extracted strip.
- **`sensor_center`** – Tuple[int, int] Coordinates of the center of the sensor (x, y).
- **`sensor_size`** – Tuple[int, int] Size of the sensor (width, height).
- **`sensor_search_area`** – Tuple[int, int] Size of the sensor search area (width, height).
- **`peak_expected_relative_location`** – list[float, …] List of expected relative peak (band) positions in the sensor (0.0 -> 1.0).
- **`control_band_index`** – int Index of the control band in the peak_expected_relative_location. (Optional, default -1 := right-most)
- **`min_control_bar_width`** – int Minimum width of the control bar (in pixels). (Optional, default 7)

Returns Realigned sensor: np.ndarray

Returns Sensor coordinates: [y0, y, x0, x]

Returns sensor_score: score for the sensor extracted (obsolete: fixed at 1.0)

Return type tuple

`pypocquant.lib.analysis.`**`extract_rotated_strip_from_box`**(*box_gray*, *box*)

Segments the strip from the box image and rotates it so that it is horizontal.

Parameters

- **`box_gray`** – Gray image of QR code box containing strip
- **`box`** – RGB image of QR code box containing strip

Returns strip_gray Extracted gray strip from box

Returns strip Extracted RGB strip from box

`pypocquant.lib.analysis.`**`adapt_bounding_box`**(*bw*, *x0*, *y0*, *width*, *height*, *fraction=0.75*)

Make the bounding box come closer to the strip by remove bumps along the outline.

Parameters

- **`bw`** – Binary mask of an image.
- **`x0`** – Top left corner in x.
- **`y0`** – Top left corner in y
- **`width`** – Mask width
- **`height`** – Mask height
- **`fraction`** –

Returns new_y0:

Returns new_y:

Returns new_x0:

Returns new_x:

`pypocquant.lib.analysis.`**`point_in_rect`**(*point*, *rect*)

Check if the given point (x, y) is contained in the rect (x0, y0, width, height).

Parameters **`point`** – Point to be checked if in rectangle

:param rect Rectangle defined by (x0, y0, width, height)

returns bool: :rtype: bool

`pypocquant.lib.analysis.`**`get_rectangles_from_image_and_rectangle_props`**(*img_shape*, *rectangle_props=(0.52, 0.15, 0.09)*)

Calculate the left and right rectangles to be used for the orientation analysis using the Hough transform.

Parameters **`img_shape`** – tuple

Image shape (width, height)

Parameters **`rectangle_props`** – tuple Tuple containing information about the relative position of the two rectangles to be searched for the inlet on both sides of the center of the image:

rectangle_props[0]: relative (0..1) vertical height of the rectangle with respect to the image height.

rectangle_props[1]: relative distance of the left edge of the right rectangle with respect to the center of the image.

rectangle_props[2]: relative distance of the left edge of the left rectangle with respect to the center of the image.

Returns left_rect: Left rectangles

Returns right_rect: Right rectangles

Return type tuple

`pypocquant.lib.analysis.`**`use_hough_transform_to_rotate_strip_if_needed`**(*img_gray*, *rectangle_props=(0.52, 0.15, 0.09)*, *stretch=False*, *img=None*, *qc=False*)

Estimate the orientation of the strip looking at features in the area around the expected sensor position. If the orientation is estimated to be wrong, rotate the strip.

Parameters

- **`img_gray`** – np.ndarray Gray-scale image to be analyzed.
- **`rectangle_props`** – tuple Tuple containing information about the relative position of the two rectangles to be searched for the inlet on both sides of the center of the image:

rectangle_props[0]: relative (0..1) vertical height of the rectangle with respect to the image height.

rectangle_props[1]: relative distance of the left edge of the right rectangle with respect to the center of the image.

rectangle_props[2]: relative distance of the left edge of the left rectangle with respect to the center of the image.

- **stretch** – bool Set to True to apply auto-stretch to the image for Hough detection (1, 99 percentile). The *original* image will be rotated, if needed.
- **img** – np.ndarray or None (default) Apply correction also to this image, if passed.
- **qc** – bool If True, create quality control images.

Returns img_gray: Gray image.

Returns img: RGB Image.

Returns qc_image QC imageg.

Returns rotated Bool; true if was rotated

Returns left_rect: Left rectangles

Returns right_rect: Right rectangles

Return type tuple

pypocquant.lib.analysis.**use_ocr_to_rotate_strip_if_needed**(*img_gray*, *img=None*, *text='COVID'*, *on_right=True*)

Try reading the given text on the strip. The text is expected to be on one side of the strip; if it is found on the other side, rotate the strip.

We apply the same rotation also to the second image, if passed.

Parameters

- **img_gray** – Gray input image to be potentially rotated.
- **img** – RGB input image to be potentially rotated.
- **text** – Text to be identified by OCR.
- **on_right** – Position of text to be identified in respect to the strip orientation.

Returns img_gray: Gray image.

Returns img: RGB Image.

Returns rotated Bool; true if was rotated

pypocquant.lib.analysis.**read_patient_data_by_ocr**(*image*, *known_manufacturers=consts.KnownManufacturers*)

Try to extract the patient data by OCR.

Parameters

- **image** – Input image to be read with OCR.
- **known_manufacturers** – List with known manufacturers.

Returns fid: FID number.

Returns manufacturer: manufacturer name.

pypocquant.lib.barcode

Module Contents

Classes

Barcode	Pythonic barcode object.

Functions

detect(image: np.ndarray, expected_area=22000, expected_aspect_ratio=7.5, barcode_border=75, blur_size=(3, 3), morph_rect=(9, 3), mm_iter=1, qc=True, verbose=False)	Detect the barcode in the image.
rotate(image, angle)	Rotate the image by given angle in degrees.
calc_area_and_approx_aspect_ratio(contour)	Calculate area and approximate aspect ratio of a contour.
rotate_90_if_needed(image)	Try to estimate the orientation of the image, and rotate if needed.
read_FID_from_barcode_image(image)	Read the FID string from the barcode image using pytesseract and
get_fid_from_barcode_data(barcode_data, barcode_type='CODE128')	Parse the output of pyzbar and retrieve the FID.
get_fid_from_box_image_using_ocr(box_img)	Use pytesseract to retrieve FID from the strip box image.
try_extracting_barcode_from_box_with_rotation(box, scaling=(1.0, 0.5, 0.25), verbose=False, log_list=None)	Try extracting barcode from QR code box while scaling it for different orientations [0, 90, 180, -90].
try_extracting_barcode_with_rotation(image, angle_range=15, verbose=True, log_list: list = None)	Try extracting barcode from QR code box for a list of angles in the range of *angle_range*.
find_strip_box_from_barcode_data_fh(image, barcode_data, qr_code_border=30, qc=False)	Extract the box around the strip using the QR barcode data.
find_strip_box_from_barcode_data(image, barcode_data, qr_code_border=30, qr_code_spacer=40, barcode_border=80, qc=False)	Extract the box around the strip using the QR barcode data.
try_extracting_barcode_with_linear_stretch(image, lower_bound_range=(25,), upper_bound_range=(98,))	Try to extract the barcodes from the image by rescaling the intensity of the image with a linear stretch.
try_getting_fid_from_code128_barcode(barcode_data)	Try finding a CODE 128 barcode in barcode data that should contain the patient FID.
try_get_fid_from_rgb(image)	Extract FID from rgb image.
try_extracting_fid_and_all_barcodes_with_linear_stretch_fh(image, lower_bound_range=(0, 5, 15, 25, 35), upper_bound_range=(100, 98, 95, 92, 89), scaling=(1.0,))	Try extracting the fid and all barcodes from the image by rescaling the intensity of the image with a
try_extracting_all_barcodes_with_linear_stretch(image, lower_bound_range=(0, 5, 15, 25, 35), upper_bound_range=(100, 98, 95, 92, 89))	Try extracting the fid and all barcodes from the image by rescaling the intensity of the image with a
rotate_if_needed_fh(image, barcode_data, image_log, verbose=True)	Rotate the image if the orientation is not the expected one.

continues on next page

Table 3 – continued from previous page

rotate_if_needed(image, barcode_data, image_log, verbose=True)	Rotate the image if the orientation is not the expected one.
pick_FID_from_candidates(fid_pyzbar, fid_tesseract)	Selection of FID from candidates depending on if candidates contain a FID.
mask_strip(strip_gray, x_barcode, qr_code_extents)	Hide the barcode on the strip image.
extract_strip_from_box(box, qr_code_width, qr_code_height, qr_code_spacer=40, slack=0)	Extract the strip from the strip box.
get_fid_numeric_value_fh(fid)	Return the numeric value of the FID (as string).
get_fid_numeric_value(fid)	Return the numeric value of the FID.
get_box_rotation_angle(pt1, pt2, pt3)	Determine the the QR code box rotation angle
align_box_with_image_border_fh(barcode_data, image)	Method to align QR code box with image border of the full image (old pipeline).
align_box_with_image_border(barcode_data, image)	Method to align QR code box with image border of the full image.

class pypocquant.lib.barcode.**Barcode**(*top: int*, *left: int*, *width: int*, *height: int*, *data: Union[bytes, str]*, *symbol: str*)

Bases: object

Pythonic barcode object.

classmethod from_barcode(*cls*, *barcode*)

Initialize from pyzbar barcode object.

Parameters barcode – A barcode (QR, CODE39, CODE128).

scale(*self*, *factor: float*)

Scale the barcode object by given factor.

The (top, left) is scaled accordingly.

Parameters factor – Scaling factor for the barcode.

__str__(*self*)

Return str(self).

__repr__(*self*)

Return repr(self).

pypocquant.lib.barcode.**detect**(*image: np.ndarray*, *expected_area=22000*, *expected_aspect_ratio=7.5*, *barcode_border=75*, *blur_size=(3, 3)*, *morph_rect=(9, 3)*, *mm_iter=1*, *qc=True*, *verbose=False*)

Detect the barcode in the image.

Adapted from: https://www.pyimagesearch.com/2014/11/24/detecting-barcodes-images-python-opencv/

Returns the extracted barcode image, the coordinates of the extracted rectangle, the (possibly rotated) image, and (if qc is True) a copy of the (possibly rotated) image with the extracted rectangle coordinates overlaid on it.

Parameters

- **image** (*np.ndarray*) – Image from which barcode should be read
- **expected_area** (*int*) – Expected area for barcode.
- **expected_aspect_ratio** (*float*) – Aspect ratio for barcode.
- **barcode_border** (*int*) – Border of the barcode.
- **blur_size** (*tuple*) – Kernel (3,3) by default for bluring the image.

- **morph_rect** (*tuple*) – Kernel (9,3) by default for morph rect.
- **mm_iter** (*int*) – Dilation & Eroding iterations.
- **qc** (*bool*) – Bool, if true quality control images will be saved.
- **verbose** (*bool*) – Bool, if true additional loggin info will be displayed.

Returns barcode_img: The image of the barcode.

Returns coordinates: The position and size coordinates of the barcode (x,y, w, h)/

Return type coordinates: tuple

Returns image: The image.

Returns mask_image The mask of the image.

Return type tuple

`pypocquant.lib.barcode.`**`rotate`**(*image*, *angle*)

Rotate the image by given angle in degrees.

Parameters

- **image** – The image to be rotated.
- **angle** – Rotation angle in degrees for the image.

Returns image: Rotated image.

`pypocquant.lib.barcode.`**`calc_area_and_approx_aspect_ratio`**(*contour*)

Calculate area and approximate aspect ratio of a contour.

Parameters **contour** – cv2.Contour.

Returns area: Area of the contour.

Returns aspect_ratio: Aspect ratio of the contour.

`pypocquant.lib.barcode.`**`rotate_90_if_needed`**(*image*)

Try to estimate the orientation of the image, and rotate if needed.

@TODO: This is not very robust so far.

Parameters **image** – Image to be rotated by 90 degrees.

Returns image: By 90 degrees rotated image.

`pypocquant.lib.barcode.`**`read_FID_from_barcode_image`**(*image*)

Read the FID string from the barcode image using pytesseract and decode the barcode itself using pyzbar.

Parameters **image** – Image to read FID from barcode.

Returns fid_tesseract: FID detected by tesseract (OCR).

Returns fid_pyzbar: FID detected by pyzbar (barcode).

Returns score: Score how well FID detection worked. For more details about the score read the manual.

`pypocquant.lib.barcode.`**`get_fid_from_barcode_data`**(*barcode_data*, *barcode_type='CODE128'*)

Parse the output of pyzbar and retrieve the FID.

Parameters

- **barcode_data** – Barcode data (zbar).
- **barcode_type** – Type of barcode (CODE39, CODE128, QRCODE).

Returns barcode: Decoded barcode as utf8.

`pypocquant.lib.barcode.`**`get_fid_from_box_image_using_ocr`**(*box_img*)

Use pytesseract to retrieve FID from the strip box image.

Parameters **`box_img`** – Image of the QR code box.

Returns fid_tesseract: FID detected by tesseract from image using OCR.

`pypocquant.lib.barcode.`**`try_extracting_barcode_from_box_with_rotations`**(*box, scaling=(1.0, 0.5, 0.25), verbose=False, log_list=None*)

Try extracting barcode from QR code box while scaling it for different orientations [0, 90, 180, -90].

Parameters

- **`box`** – QR code box
- **`scaling`** – Scaling factors.
- **`verbose`** – Display additional logging information to the console.
- **`log_list`** – Log list.

Returns fid: FID number

Returns log_list Appended Log list with current log information.

`pypocquant.lib.barcode.`**`try_extracting_barcode_with_rotation`**(*image, angle_range=15, verbose=True, log_list: list = None*)

Try extracting barcode from QR code box for a list of angles in the range of *angle_range*.

Parameters

- **`image`** – Input image
- **`angle_range`** (*`int`*) – Range of angles to rotate input images in degrees.
- **`verbose`** – Display additional logging information to the console.
- **`log_list`** – Log list.

Returns fid: Extracted FID

Returns angle: Rotation angle that led to FID detection

Returns log_list: Appended log list.

`pypocquant.lib.barcode.`**`find_strip_box_from_barcode_data_fh`**(*image, barcode_data, qr_code_border=30, qc=False*)

Extract the box around the strip using the QR barcode data.

Parameters

- **`image`** – Strip image.
- **`barcode_data`** – Barcode data.

- **qr_code_border** – Border around QR codes.
- **qc** – Bool, if true quality control image will be saved.

Returns box: Strip box.

Returns qr_code_size: The size of the QR codes (qr_code_width, qr_code_height).

Returns qc_image: Quality control image.

Returns box_rect: Rectangle of the QR box.

pypocquant.lib.barcode.**find_strip_box_from_barcode_data**(*image*, *barcode_data*, *qr_code_border=30*, *qr_code_spacer=40*, *barcode_border=80*, *qc=False*)

Extract the box around the strip using the QR barcode data.

Parameters

- **image** – Input image.
- **barcode_data** – Barcode data
- **qr_code_border** – Border around QR code on image.
- **qr_code_spacer** – Spacer around QR code.
- **barcode_border** – Border around barcode such as CODE128.
- **qc** – Bool, if true quality control image will be saved.

Returns box QR code box around strip

Returns x_barcode Return the (x) coordinate of the left edge of the barcode rectangle.

Returns qr_code_size: The size of the QR codes (qr_code_width, qr_code_height).

Returns qc_image Quality control image.

pypocquant.lib.barcode.**try_extracting_barcode_with_linear_stretch**(*image*, *lower_bound_range=(25)*, *upper_bound_range=(98)*)

Try to extract the barcodes from the image by rescaling the intensity of the image with a linear stretch.

Parameters

- **image** – Input image
- **lower_bound_range** – Lower bound range.
- **lower_bound_range** – tuple
- **upper_bound_range** – Upper bound range.
- **upper_bound_range** – tuple

Returns

””

Returns gray

pypocquant.lib.barcode.**try_getting_fid_from_code128_barcode**(*barcode_data*)

Try finding a CODE 128 barcode in barcode data that should contain the patient FID.

Parameters **barcode_data** – Barcode data

Returns barcode: Decoded CODE128 barcode.

`pypocquant.lib.barcode.`**`try_get_fid_from_rgb`**(*image*)

Extract FID from rgb image.

Parameters **`image`** – RGB image with FID.

Returns fid: Detected FID as string.

`pypocquant.lib.barcode.`**`try_extracting_fid_and_all_barcodes_with_linear_stretch_fh`**(*image*, *lower_bound*
5,
15,
25,
35), *up-*
per_bound_r
98,
95,
92,
89), *scal-*
ing=(1.0))

Try extracting the fid and all barcodes from the image by rescaling the intensity of the image with a linear stretch.

Parameters

- **`image`** – Input image
- **`lower_bound_range`** – Lower bound range.
- **`lower_bound_range`** – tuple
- **`upper_bound_range`** – Upper bound range.
- **`upper_bound_range`** – tuple
- **`scaling`** – Scaling factor
- **`scaling`** – tuple

Returns barcodes: Barcode object

Returns fid: FID number

Returns manufacturer: Manufacturer name.

Returns plate: Plate info.

Returns well: Well info.

Returns user: Additional user data.

Returns best_lb: Best lower bound.

Returns best_ub: Best upper bound

Returns best_score: Best score.

Returns best_scaling_factor: Best scaling factor

Returns fid_128: FID 128 code.

pypocquant.lib.barcode.**try_extracting_all_barcodes_with_linear_stretch**(*image*, *lower_bound_range=(0, 5, 15, 25, 35)*, *upper_bound_range=(100, 98, 95, 92, 89)*)

Try extracting the fid and all barcodes from the image by rescaling the intensity of the image with a linear stretch.

Parameters

- **image** – Input image.
- **lower_bound_range** – Lower bound range.
- **lower_bound_range** – tuple
- **upper_bound_range** – Upper bound range.
- **upper_bound_range** – tuple

Returns best_barcode_data

Returns best_lb

Returns best_ub

Returns best_score

pypocquant.lib.barcode.**rotate_if_needed_fh**(*image*, *barcode_data*, *image_log*, *verbose=True*)

Rotate the image if the orientation is not the expected one.

Parameters

- **image** – Input image.
- **barcode_data** – Barcode data.
- **image_log** – Image log list.
- **verbose** (*bool*) – Bool, if true displays additional information to the console.

Returns image_was_rotated: Bool, true if image was rotated.

Return type image_was_rotated: bool

Returns image: Rotated image.

Return type tuple

pypocquant.lib.barcode.**rotate_if_needed**(*image*, *barcode_data*, *image_log*, *verbose=True*)

Rotate the image if the orientation is not the expected one.

Parameters

- **image** – Input image.
- **barcode_data** – Barcode data.
- **image_log** – Image log list.

- **verbose** (*bool*) – Bool, if true displays additional information to the console.

Returns image_was_rotated: Bool, true if image was rotated.

Return type image_was_rotated: bool

Returns image: Rotated image.

Returns image_log: Log for this image

Return type tuple

pypocquant.lib.barcode.**pick_FID_from_candidates**(*fid_pyzbar*, *fid_tesseract*)
Selection of FID from candidates depending on if candidates contain a FID.

Parameters

- **fid_pyzbar** – FID string determined with pyzbar.
- **fid_tesseract** – FID string determined with tesseract.

Returns fid FID number

Returns score Score for the candidate determination.

pypocquant.lib.barcode.**mask_strip**(*strip_gray*, *x_barcode*, *qr_code_extents*)
Hide the barcode on the strip image.

Parameters

- **strip_gray** – Image of the strip (POCT).
- **x_barcode** – X coordinate of the barcode on the strip.
- **qr_code_extents** – QR code extents on the strip.

Returns strip_gray_masked Strip with QR code masked away.

Returns background_value Background value used for strip masking.

pypocquant.lib.barcode.**extract_strip_from_box**(*box*, *qr_code_width*, *qr_code_height*, *qr_code_spacer=40*, *slack=0*)
Extract the strip from the strip box.

Parameters

- **box** – Image of the QR code box.
- **qr_code_width** – Width ot the QR code
- **qr_code_height** – Height ot the QR code
- **qr_code_spacer** – Horizontal and vertical distance between the internal edge of the QR codes and the beginning of the strip.
- **slack** – Some buffer (subtracted from qr_code_spacer) to avoid cropping into the strip

Returns strip Returns the extracted POCT strip as image matrix.

pypocquant.lib.barcode.**get_fid_numeric_value_fh**(*fid*)
Return the numeric value of the FID (as string).

A FID could be in the form 'F0123456'. We want to preserve the leading 0 after we removed the 'F'.

Parameters **fid** (*str*) – FID number

Returns fid:

pypocquant.lib.barcode.**get_fid_numeric_value**(*fid*)
Return the numeric value of the FID.

Parameters **fid** (*str*) – FID number

Returns filtered_fid: FID number as numeric

`pypocquant.lib.barcode.`**`get_box_rotation_angle`**(*pt1*, *pt2*, *pt3*)

Determine the the QR code box rotation angle

Parameters

- **pt1** – Coordinate corner 1
- **pt2** – Coordinate corner 2
- **pt3** – Coordinate corner 3

Returns rot_angle Rotation angle in degree.

`pypocquant.lib.barcode.`**`align_box_with_image_border_fh`**(*barcode_data*, *image*)

Method to align QR code box with image border of the full image (old pipeline).

Parameters

- **barcode_data** – QR code data
- **image** – Image

Returns image_rotated: Rotated image

Returns angle Rotation angle in degrees.

`pypocquant.lib.barcode.`**`align_box_with_image_border`**(*barcode_data*, *image*)

Method to align QR code box with image border of the full image.

Parameters

- **barcode_data** – QR code data
- **image** – Image

Returns image_rotated: Rotated image

Returns angle Rotation angle in degrees.

pypocquant.lib.consts

Module Contents

Classes

Issue	Issues detected during image processing.
SymbolTypes	SymbolTypes of zbar. Currently we support CODE39, CODE128 and QRCODE detection.

class `pypocquant.lib.consts.`**`Issue`**

Bases: `enum.Enum`

Issues detected during image processing.

Returns Issue code

NONE = 0

BARCODE_EXTRACTION_FAILED = 1

FID_EXTRACTION_FAILED = 2

STRIP_BOX_EXTRACTION_FAILED = 3

STRIP_EXTRACTION_FAILED = 4

POOR_STRIP_ALIGNMENT = 5

SENSOR_EXTRACTION_FAILED = 6

BAND_QUANTIFICATION_FAILED = 7

CONTROL_BAND_MISSING = 8

class pypocquant.lib.consts.**SymbolTypes**

Bases: enum.Enum

SymbolTypes of zbar. Currently we support CODE39, CODE128 and QRCODE detection. :param Enum:

Returns TYPES

Return type list

TYPES

pypocquant.lib.consts.**KnownManufacturers = ['AUGURIX', 'BIOZAK', 'CTKBIOTECH', 'DRALBERMEXA**

pypocquant.lib.consts.**BAND_COLORS**

pypocquant.lib.io

Module Contents

Functions

load_and_process_image(full_filename: str, raw_auto_stretch: bool = False, raw_auto_wb: bool = False, to_rgb: bool = False)	Load a supported (standard) image file format such as '.jpg', '.tif', '.png' and
is_raw(filename: str) → bool	Check whether the image is one of the supported RAW images

pypocquant.lib.io.**load_and_process_image** (*full_filename: str, raw_auto_stretch: bool = False, raw_auto_wb: bool = False, to_rgb: bool = False*)

Load a supported (standard) image file format such as '.jpg', '.tif', '.png' and some RAW file formats ('.nef', '.cr2', .'arw').

Parameters

- **full_filename** (*str*) – Full path to the file to open.
- **raw_auto_stretch** (*bool*) – (Only applies to RAW image file formats). Set to True to automatically stretch image intensities (default = False).
- **raw_auto_wb** (*bool*) – (Only applies to RAW image file formats). Set to True to automatically apply white-balancing (default = False).
- **to_rgb** (*bool*) – Set to True to convert from BGR (openCV standard, used in processing) to RGB (for display, default = False).

Returns image: Loaded (and possibly processed) image, or None it the image could not be opened.

Return type cv2.Image

`pypocquant.lib.io.`**`is_raw`**(*filename: str*) → bool

Check whether the image is one of the supported RAW images (by checking the file extension.

Parameters **`filename`** (*str*) – Full file name.

Returns bool True if the image is RAW, false otherwise.

Return type bool

pypocquant.lib.pipeline

Module Contents

Functions

run_pool(files, raw_auto_stretch, raw_auto_wb, input_folder_path, results_folder_path, strip_try_correct_orientation, strip_try_correct_orientation_rects, strip_text_to_search, strip_text_on_right, min_sensor_score, qr_code_border, perform_sensor_search, sensor_size, sensor_center, sensor_search_area, sensor_thresh_factor, sensor_border, peak_expected_relative_location, control_band_index, subtract_background, force_fid_search, sensor_band_names, verbose, qc, max_workers=4)	Run a thread pool for the analysis.
run_pipeline(input_folder_path: Path, results_folder_path: Path, raw_auto_stretch: bool = False, raw_auto_wb: bool = False, strip_try_correct_orientation: bool = False, strip_try_correct_orientation_rects: tuple = (0.52, 0.15, 0.09), strip_text_to_search: str = 'COVID', strip_text_on_right: bool = True, min_sensor_score: float = 0.85, qr_code_border: int = 30, perform_sensor_search: bool = True, sensor_size: tuple = (61, 249), sensor_center: tuple = (178, 667), sensor_search_area: tuple = (71, 259), sensor_thresh_factor: float = 2, sensor_border: tuple = (7, 7), peak_expected_relative_location: tuple = (0.25, 0.53, 0.79), control_band_index: int = -1, subtract_background: bool = True, force_fid_search: bool = False, sensor_band_names: tuple = ('igm', 'igg', 'ctl'), verbose: bool = False, qc: bool = False, max_workers: int = 2)	Run the whole processing and analysis pipeline.

continues on next page

Table 6 – continued from previous page

run(filename, raw_auto_stretch, raw_auto_wb, input_folder_path: Path, results_folder_path: Path, strip_try_correct_orientation: bool, strip_try_correct_orientation_rects: tuple, strip_text_to_search: str, strip_text_on_right: bool, min_sensor_score: float = 0.85, qr_code_border: int = 30, perform_sensor_search: bool = False, sensor_size: tuple = (37, 185), sensor_center: tuple = (119, 471), sensor_search_area: tuple = (50, 195), sensor_thresh_factor: float = 2, sensor_border: tuple = (7, 7), peak_expected_relative_location: tuple = (0.27, 0.55, 0.79), control_band_index: int = -1, subtract_background: bool = True, force_fid_search: bool = False, sensor_band_names: tuple = ('igm', 'igg', 'ctl'), verbose: bool = False, qc: bool = False)	Runnable which runs the analysis on a worker

`pypocquant.lib.pipeline.`**`run_pool`**(*files*, *raw_auto_stretch*, *raw_auto_wb*, *input_folder_path*, *results_folder_path*, *strip_try_correct_orientation*, *strip_try_correct_orientation_rects*, *strip_text_to_search*, *strip_text_on_right*, *min_sensor_score*, *qr_code_border*, *perform_sensor_search*, *sensor_size*, *sensor_center*, *sensor_search_area*, *sensor_thresh_factor*, *sensor_border*, *peak_expected_relative_location*, *control_band_index*, *subtract_background*, *force_fid_search*, *sensor_band_names*, *verbose*, *qc*, *max_workers=4*)

Run a thread pool for the analysis.

Parameters

- **files** (*list*) – List with image file names to be processed
- **raw_auto_stretch** (*bool*) – Whether to automatically correct the white balance of RAW images on load. This does not affect JPEG images!
- **raw_auto_wb** (*bool*) – Whether to automatically stretch image intensities of RAW images on load. This does not affect JPEG images!
- **input_folder_path** (*str*) – Folder with the raw images to process.
- **results_folder_path** (*str*) – Target folder, where all results and quality control figures are written.
- **strip_try_correct_orientation** (*bool*) – Try to assess and possibly correct for wrong orientation of the strip by searching for the position of the injection inlet.
- **strip_try_correct_orientation_rects** (*tuple*) – Tuple containing information about the relative position of the two rectangles to be searched for the inlet on both sides of the center of the image:

  rectangle_props[0]: relative (0..1) vertical height of the rectangle with respect to the image height.

  rectangle_props[1]: relative (0..1) distance of the left edge of the right rectangle with respect to the center of the image.

  rectangle_props[2]: relative (0..1) distance of the left edge of the left rectangle with respect to the center of the image.

- **strip_text_to_search** (*str*) – Text to search on the strip to assess orientation. Set to "" to skip.
- **strip_text_on_right** (*bool*) – Assuming the strip is oriented horizontally, whether the 'strip_text_to_search' text is assumed to be on the right. If 'strip_text_on_right' is True and the text is found on the left hand-side of the strip, the strip will be rotated 180 degrees. Ignored if strip_text_to_search is "".
- **min_sensor_score** (*float*) – Minimum segmentation score for the sensor to be considered peak analysis (0.0 <= min_sensor_score <= 1.0). **This is currently ignored.**
- **qr_code_border** (*int*) – Lateral and vertical extension of the (white) border around each QR code.
- **perform_sensor_search** (*bool*) – If True, the (inverted) sensor is searched within 'sensor_search_area' around the expected 'sensor_center'; if False, the sensor of size 'sensor_size' is simply extracted from the strip image centered at the relative strip position 'sensor_center'.
- **sensor_size** (*tuple*) – Area of the sensor to be extracted (height, width).
- **sensor_center** (*tuple*) – Coordinates of the center of the sensor with respect to the strip image (y, x).
- **sensor_search_area** (*tuple*) – Search area around the sensor (height, width). Used only if 'skip_sensor_search' is False.
- **sensor_thresh_factor** (*int*) – Set the number of (robust) standard deviations away from the median band background for a peak to be considered valid.
- **sensor_border** (*tuple*) – Lateral and vertical sensor border to be ignored in the analysis to avoid border effects.
- **peak_expected_relative_location** (*tuple*) – Expected relative peak positions as a function of the width of the sensor (= 1.0)
- **control_band_index** (*int*) – Index of the control band in the peak_expected_relative_location. (Optional, default -1 := right-most)
- **subtract_background** (*bool*) – If True, estimate and subtract the background of the sensor intensity profile.
- **force_fid_search** (*bool*) – If True, apply a series of search fall-back approaches to extract patient data from the image. Only use this if the expected QR code with patient data was not added to the image or could not be extracted.
- **sensor_band_names** (*tuple*) – Names of the bands for the data frame header. Please notice: the third ([2]) band is always the control band.
- **verbose** (*bool*) – Toggle verbose output.
- **qc** (*bool*) – Toggle creation of quality control figures.
- **max_workers** (*int*) – Number of max cores to use for running the pipeline

Returns res

Return type list

Returns log_list

Return type list

pypocquant.lib.pipeline.**run_pipeline**(*input_folder_path: Path*, *results_folder_path: Path*, *raw_auto_stretch: bool = False*, *raw_auto_wb: bool = False*, *strip_try_correct_orientation: bool = False*, *strip_try_correct_orientation_rects: tuple = (0.52, 0.15, 0.09)*, *strip_text_to_search: str = 'COVID'*, *strip_text_on_right: bool = True*, *min_sensor_score: float = 0.85*, *qr_code_border: int = 30*, *perform_sensor_search: bool = True*, *sensor_size: tuple = (61, 249)*, *sensor_center: tuple = (178, 667)*, *sensor_search_area: tuple = (71, 259)*, *sensor_thresh_factor: float = 2*, *sensor_border: tuple = (7, 7)*, *peak_expected_relative_location: tuple = (0.25, 0.53, 0.79)*, *control_band_index: int = - 1*, *subtract_background: bool = True*, *force_fid_search: bool = False*, *sensor_band_names: tuple = ('igm', 'igg', 'ctl')*, *verbose: bool = False*, *qc: bool = False*, *max_workers: int = 2*)

Run the whole processing and analysis pipeline.

Parameters

- **input_folder_path** (*str*) – Folder with the raw images to process.
- **results_folder_path** (*str*) – Target folder, where all results and quality control figures are written.
- **raw_auto_stretch** (*bool*) – Whether to automatically correct the white balance of RAW images on load. This does not affect JPEG images!
- **raw_auto_wb** (*bool*) – Whether to automatically stretch image intensities of RAW images on load. This does not affect JPEG images!
- **strip_try_correct_orientation** (*bool*) – Try to assess and possibly correct for wrong orientation of the strip by searching for the position of the injection inlet.
- **strip_try_correct_orientation_rects** (*tuple*) – Tuple containing information about the relative position of the two rectangles to be searched for the inlet on both sides of the center of the image:

  rectangle_props[0]: relative (0..1) vertical height of the rectangle with respect to the image height.

  rectangle_props[1]: relative (0..1) distance of the left edge of the right rectangle with respect to the center of the image.

  rectangle_props[2]: relative (0..1) distance of the left edge of the left rectangle with respect to the center of the image.
- **strip_text_to_search** (*str*) – Text to search on the strip to assess orientation. Set to “” to skip.
- **strip_text_on_right** (*bool*) – Assuming the strip is oriented horizontally, whether the ‘strip_text_to_search’ text is assumed to be on the right. If ‘strip_text_on_right’ is True and the text is found on the left hand-side of the strip, the strip will be rotated 180 degrees. Ignored if strip_text_to_search is “”.
- **min_sensor_score** (*float*) – Minimum segmentation score for the sensor to be considered peak analysis (0.0 <= min_sensor_score <= 1.0). **This is currently ignored.**
- **qr_code_border** (*int*) – Lateral and vertical extension of the (white) border around each QR code.

- **perform_sensor_search** (*bool*) – If True, the (inverted) sensor is searched within 'sensor_search_area' around the expected 'sensor_center'; if False, the sensor of size 'sensor_size' is simply extracted from the strip image centered at the relative strip position 'sensor_center'.
- **sensor_size** (*tuple*) – Area of the sensor to be extracted (height, width).
- **sensor_center** (*tuple*) – Coordinates of the center of the sensor with respect to the strip image (y, x).
- **sensor_search_area** (*tuple*) – Search area around the sensor (height, width). Used only if 'skip_sensor_search' is False.
- **sensor_thresh_factor** (*int*) – Set the number of (robust) standard deviations away from the median band background for a peak to be considered valid.
- **sensor_border** (*tuple*) – Lateral and vertical sensor border to be ignored in the analysis to avoid border effects.
- **peak_expected_relative_location** (*tuple*) – Expected relative peak positions as a function of the width of the sensor (= 1.0)
- **control_band_index** (*int*) – Index of the control band in the peak_expected_relative_location. (Optional, default -1 := right-most)
- **subtract_background** (*bool*) – If True, estimate and subtract the background of the sensor intensity profile.
- **force_fid_search** (*bool*) – If True, apply a series of search fall-back approaches to extract patient data from the image. Only use this if the expected QR code with patient data was not added to the image or could not be extracted.
- **sensor_band_names** (*tuple*) – Names of the bands for the data frame header. Please notice: the third ([2]) band is always the control band.
- **verbose** (*bool*) – Toggle verbose output.
- **qc** (*bool*) – Toggle creation of quality control figures.
- **max_workers** (*int*) – Number of max cores to use for running the pipeline

pypocquant.lib.pipeline.**run**(*filename*, *raw_auto_stretch*, *raw_auto_wb*, *input_folder_path: Path*, *results_folder_path: Path*, *strip_try_correct_orientation: bool*, *strip_try_correct_orientation_rects: tuple*, *strip_text_to_search: str*, *strip_text_on_right: bool*, *min_sensor_score: float = 0.85*, *qr_code_border: int = 30*, *perform_sensor_search: bool = False*, *sensor_size: tuple = (37, 185)*, *sensor_center: tuple = (119, 471)*, *sensor_search_area: tuple = (50, 195)*, *sensor_thresh_factor: float = 2*, *sensor_border: tuple = (7, 7)*, *peak_expected_relative_location: tuple = (0.27, 0.55, 0.79)*, *control_band_index: int = - 1*, *subtract_background: bool = True*, *force_fid_search: bool = False*, *sensor_band_names: tuple = ('igm', 'igg', 'ctl')*, *verbose: bool = False*, *qc: bool = False*)

Runnable which runs the analysis on a worker

Parameters

- **filename** (*list*) – Image file name to be processed
- **raw_auto_stretch** (*bool*) – Whether to automatically correct the white balance of RAW images on load. This does not affect JPEG images!

- **raw_auto_wb** (*bool*) – Whether to automatically stretch image intensities of RAW images on load. This does not affect JPEG images!
- **input_folder_path** (*str*) – Folder with the raw images to process.
- **results_folder_path** (*str*) – Target folder, where all results and quality control figures are written.
- **strip_try_correct_orientation** (*bool*) – Try to assess and possibly correct for wrong orientation of the strip by searching for the position of the injection inlet.
- **strip_try_correct_orientation_rects** (*tuple*) – Tuple containing information about the relative position of the two rectangles to be searched for the inlet on both sides of the center of the image:

  rectangle_props[0]: relative (0..1) vertical height of the rectangle with respect to the image height.

  rectangle_props[1]: relative (0..1) distance of the left edge of the right rectangle with respect to the center of the image.

  rectangle_props[2]: relative (0..1) distance of the left edge of the left rectangle with respect to the center of the image.
- **strip_text_to_search** (*str*) – Text to search on the strip to assess orientation. Set to "" to skip.
- **strip_text_on_right** (*bool*) – Assuming the strip is oriented horizontally, whether the 'strip_text_to_search' text is assumed to be on the right. If 'strip_text_on_right' is True and the text is found on the left hand-side of the strip, the strip will be rotated 180 degrees. Ignored if strip_text_to_search is "".
- **min_sensor_score** (*float*) – Minimum segmentation score for the sensor to be considered peak analysis (0.0 <= min_sensor_score <= 1.0). **This is currently ignored.**
- **qr_code_border** (*int*) – Lateral and vertical extension of the (white) border around each QR code.
- **perform_sensor_search** (*bool*) – If True, the (inverted) sensor is searched within 'sensor_search_area' around the expected 'sensor_center'; if False, the sensor of size 'sensor_size' is simply extracted from the strip image centered at the relative strip position 'sensor_center'.
- **sensor_size** (*tuple*) – Area of the sensor to be extracted (height, width).
- **sensor_center** (*tuple*) – Coordinates of the center of the sensor with respect to the strip image (y, x).
- **sensor_search_area** (*tuple*) – Search area around the sensor (height, width). Used only if 'skip_sensor_search' is False.
- **sensor_thresh_factor** (*int*) – Set the number of (robust) standard deviations away from the median band background for a peak to be considered valid.
- **sensor_border** (*tuple*) – Lateral and vertical sensor border to be ignored in the analysis to avoid border effects.
- **peak_expected_relative_location** (*tuple*) – Expected relative peak positions as a function of the width of the sensor (= 1.0)
- **control_band_index** (*int*) – Index of the control band in the peak_expected_relative_location. (Optional, default -1 := right-most)

- **subtract_background** (*bool*) – If True, estimate and subtract the background of the sensor intensity profile.
- **force_fid_search** (*bool*) – If True, apply a series of search fall-back approaches to extract patient data from the image. Only use this if the expected QR code with patient data was not added to the image or could not be extracted.
- **sensor_band_names** (*tuple*) – Names of the bands for the data frame header. Please notice: the third ([2]) band is always the control band.
- **verbose** (*bool*) – Toggle verbose output.
- **qc** (*bool*) – Toggle creation of quality control figures.

Returns row_data

Returns image_log

pypocquant.lib.processing

Module Contents

Functions

phase_only_correlation(in1: np.ndarray, in2: np.ndarray) → np.ndarray	Calculate phase-only correlation of two numpy arrays.
find_position_in_image_using_phase_corr(in1: np.ndarray, in2: np.ndarray) → tuple	Uses phase-only correlation to find the coordinates in *in2* where *in1* can be found.
find_position_in_image_using_norm_xcorr(in1: np.ndarray, in2: np.ndarray) → tuple	Uses normalized cross-correlation to find the coordinates in *in2* where *in1* can be found.
correlation_coefficient(image_1, image_2)	Create the normalized correlation coefficient (scalar) of two images.
crop_image_around_position_to_size(image, y, x, size)	Crop an image to given size centered at coordinates (y, x).
create_rgb_image(red, green, blue=None)	Merge three single channels into an RGB image.
find_features(image, detector='surf', num_features=1000, hessian_threshold=10, use_latch_descriptor=False)	Find features in of a template inside a larger image.
find_position_of_template_in_image_using_descriptors(template_kps, template_des, image_kps, image_des, template_size)	Find the template in the image using the extracted feature descriptors.
register_images_opencv_features(source, target, detector='surf', use_latch_descriptor=False, perspective=True, affine=False, rigid=False, num_features=1000, hessian_threshold=10, control_image=False)	Register 2 images using image features.
apply_transformation_to_image(image, transformation_type, transformation_matrix, target_height=None, target_width=None)	Apply a transformation to an image.
display_matches(img1, img2, sel_matches, k1, k2, max_matches=None)	Displays the matches on a control image and returns it.
add_border(images: list, border: int, fill_value: int = -1) → list	Add a border to each of the images in a list and sets the border values to a given fill value.
BGR2Gray(image, to_lightness=False)	Convert a BGR image to gray or lightness.

`pypocquant.lib.processing.`**`phase_only_correlation`**(*in1: np.ndarray*, *in2: np.ndarray*) → np.ndarray

Calculate phase-only correlation of two numpy arrays.

Parameters

- **`in1`** – 2D numpy array.
- **`in2`** – 2D numpy array.

Returns 2D np.float64 numpy array.

`pypocquant.lib.processing.`**`find_position_in_image_using_phase_corr`**(*in1: np.ndarray*, *in2: np.ndarray*) → tuple

Uses phase-only correlation to find the coordinates in *in2* where *in1* can be found.

Parameters

- **`in1`** – 2D numpy array (must be strictly smaller, i.e. completely contained) in in2.
- **`in2`** – 2D numpy array.

Returns tuple with (y = row, x = column) location of the center of in1 in in2.

Return type tuple

`pypocquant.lib.processing.`**`find_position_in_image_using_norm_xcorr`**(*in1: np.ndarray*, *in2: np.ndarray*) → tuple

Uses normalized cross-correlation to find the coordinates in *in2* where *in1* can be found.

Parameters

- **`in1`** – 2D numpy array (must be strictly smaller, i.e. completely contained) in in2.
- **`in2`** – 2D numpy array.

Returns tuple with (y = row, x = column) location of the center of in1 in in2.

Return type tuple

`pypocquant.lib.processing.`**`correlation_coefficient`**(*image_1*, *image_2*)

Create the normalized correlation coefficient (scalar) of two images. :param image_1: np image1 :param image_2: np image2

Returns product:

`pypocquant.lib.processing.`**`crop_image_around_position_to_size`**(*image*, *y*, *x*, *size*)

Crop an image to given size centered at coordinates (y, x).

If the original image is too small, a cropped version will be returned.

Parameters

- **`image`** – Image to be cropped
- **`y`** – y center coordinate
- **`x`** – x center coordinate
- **`size`** – size of the crop

Returns out: Cropped image

`pypocquant.lib.processing.`**`create_rgb_image`**(*red*, *green*, *blue=None*)

Merge three single channels into an RGB image.

Parameters **`red`** – Red channel

:param green Green channel

:param blue Blue channel

Returns view: RGB image

`pypocquant.lib.processing.`**`find_features`**(*image*, *detector='surf'*, *num_features=1000*, *hessian_threshold=10*, *use_latch_descriptor=False*)

Find features in of a template inside a larger image.

Parameters

- **`image`** –
- **`detector`** –
- **`num_features`** –
- **`hessian_threshold`** –
- **`use_latch_descriptor`** –

Returns kp

Returns des

`pypocquant.lib.processing.`**`find_position_of_template_in_image_using_descriptors`**(*template_kps*, *template_des*, *image_kps*, *image_des*, *template_size*)

Find the template in the image using the extracted feature descriptors.

Parameters

- **`template_kps`** –
- **`template_des`** –
- **`image_kps`** –
- **`image_des`** –
- **`template_size`** –

Returns coordinates

Return type tuple

`pypocquant.lib.processing.register_images_opencv_features(source, target, detector='surf', use_latch_descriptor=False, perspective=True, affine=False, rigid=False, num_features=1000, hessian_threshold=10, control_image=False)`

Register 2 images using image features.

Keyword arguments: :param source: source image to be registered (must be grayscale) :param target: target image (must be grayscale) :param detector: one of "orb", "kaze", "akaze", "brisk", "surf", "sift" (default if surf) :param use_latch_descriptor: True to use the new LATCH descriptor (requires openCV 3.1), False to use the

default descriptors provided by the detectors (default is False)

Parameters

- **num_features** – number of features (used only by the "orb" and "sift" detectors, default is 1000)
- **hessian_threshold** – threshold of the hessian of the images (used only by the "surf" detector, default is 10)
- **perspective** – register the image using a perspective transformation (optional, default=True).
- **affine** – register the image using an affine transformation (optional, default=False).
- **rigid** – register the image using a rigid transformation (optional, default=False).
- **control_image** – set to True to create a quality control image (default is False).

Returns results (aligned: aligned image, M : transformation matrix, mask: mask returned by cv2.findHomography()),

Return type dict

Returns view: quality control image,

Returns source_descr: list of source descriptors,

Returns target_descr: list of target descriptors).

`pypocquant.lib.processing.apply_transformation_to_image(image, transformation_type, transformation_matrix, target_height=None, target_width=None)`

Apply a transformation to an image.

Parameters

- **image** – image to be transformed.
- **transformation_type** –

  type of transformation. One of: "perspective": register the image using a perspective transformation. "affine": register the image using an affine transformation. "rigid": register the image using a rigid transformation.
- **transformation_matrix** –

transformation matrix, must be: "perspective": (3x3) "affine": (2x3) "rigid": (2x3)

- **target_height** – (optional) number of rows of the transformed image. If not set, the transformed image will have the same size as the source image.
- **target_width** – (optional) number of columns of the transformed image. If not set, the transformed image will have the same size as the source image.

Returns transformed: transformed image.

`pypocquant.lib.processing.`**`display_matches`**(*img1*, *img2*, *sel_matches*, *k1*, *k2*, *max_matches=None*)

Displays the matches on a control image and returns it.

Parameters

- **img1** – First image
- **img2** – Second image
- **sel_matches** – Selected matches
- **k1** –
- **k2** –
- **max_matches** –

Returns view

`pypocquant.lib.processing.`**`add_border`**(*images: list*, *border: int*, *fill_value: int = - 1*) → list

Add a border to each of the images in a list and sets the border values to a given fill value.

If the fill_value is omitted, the median of all pixel intensities will taken.

Parameters

- **images** (*list*) – List of images
- **border** (*int*) – Border to be added to image
- **fill_value** – (optional) If omitted the median of all pixel intensities will taken.

Returns out: List of images with added border.

Return type list

`pypocquant.lib.processing.`**`BGR2Gray`**(*image*, *to_lightness=False*)

Convert a BGR image to gray or lightness.

Parameters

- **image** – Image to be converted
- **to_lightness** – To lightness bool

Returns 1

Return type cv2.Image

pypocquant.lib.settings

Module Contents

Functions

default_settings()	Return a dictionary containing the default settings.
load_settings(filename)	Loads settings from file and returns them in a dictionary.
save_settings(settings_dictionary, filename)	Save settings from a dictionary to file.
load_list_file(filename)	Loads list from file and returns them as list.

pypocquant.lib.settings.**default_settings**()

Return a dictionary containing the default settings.

pypocquant.lib.settings.**load_settings**(*filename*)

Loads settings from file and returns them in a dictionary.

Parameters **filename** (*str*) – Name of the settings file.

Returns settings_dictionary

Return type dict

pypocquant.lib.settings.**save_settings**(*settings_dictionary*, *filename*)

Save settings from a dictionary to file.

Parameters

- **settings_dictionary** – Settings dictionary
- **filename** (*str*) – Filename of the settings file to be saved.

pypocquant.lib.settings.**load_list_file**(*filename*)

Loads list from file and returns them as list.

Parameters **filename** (*str*) – Filename of the settings file to be loaded.

Returns file_content_list

Return type list

pypocquant.lib.tools

Module Contents

Functions

extract_strip(image, qr_code_border, strip_try_correct_orientation, strip_try_correct_orientation_rects=(0.52, 0.15, 0.09), stretch_for_hough=False, strip_text_to_search='', strip_text_on_right=True)	Attempts to extract the strip from the original image.

`pypocquant.lib.tools.`**`extract_strip`**(*image*, *qr_code_border*, *strip_try_correct_orientation*, *strip_try_correct_orientation_rects=(0.52, 0.15, 0.09)*, *stretch_for_hough=False*, *strip_text_to_search=''*, *strip_text_on_right=True*)

Attempts to extract the strip from the original image.

Parameters

- **`image`** (*`numpy array`*) – RGB image to be processed.
- **`qr_code_border`** (*`int`*) – Lateral and vertical extension of the (white) border around each QR code.
- **`strip_try_correct_orientation`** (*`bool`*) – Try to assess and possibly correct for wrong orientation of the strip by searching for the position of the injection inlet.
- **`strip_try_correct_orientation_rects`** (*`tuple`*) – Tuple containing information about the relative position of the two rectangles to be searched for the inlet on both sides of the center of the image:

  rectangle_props[0]: relative (0..1) vertical height of the rectangle with respect to the image height.

  rectangle_props[1]: relative (0..1) distance of the left edge of the right rectangle with respect to the center of the image.

  rectangle_props[2]: relative (0..1) distance of the left edge of the left rectangle with respect to the center of the image.
- **`stretch_for_hough`** (*`bool (default, False)`*) – Set to True to apply auto-stretch to the image for Hough detection (1, 99 percentile).
- **`strip_text_to_search`** (*`str`*) – str Text to search on the strip to assess orientation. Set to "" to skip.
- **`strip_text_on_right`** (*`bool`*) – Assuming the strip is oriented horizontally, whether the 'strip_text_to_search' text is assumed to be on the right. If 'strip_text_on_right' is True and the text is found on the left hand-side of the strip, the strip will be rotated 180 degrees. Ignored if strip_text_to_search is "".

Returns strip_for_analysis: Strip image (RGB) or None if extraction fails.

Returns error_msg: If strip is None, the cause of failure will be stored in error_message.

Returns left_rect: Detected Hough circles in left_rect.

Returns right_rect: Detected Hough circles in right_rect.

Return type tuple

`pypocquant.lib.utils`

Module Contents

Functions

create_quality_control_images(results_folder_path: str, basename: str, map_of_images: dict, extension: str = '.png', quality: int = 100)	Save the list of requested quality control images.
get_project_root() → Path	Returns project root folder.
get_data_folder() → Path	Returns the value of the environment variable DATA_FOLDER or,
image_format_converter(directory, filename, output_dir=None, image_format='tif')	Converts a image in raw format (.'nef') to the specified open format. Default is '.tif'.
get_iso_date_from_image(image_path)	Returns the date in iso-date format for the image at the given path.
get_exif_details(image_path)	Returns the Exif metadata for the image at the given path. In particular EXIF ExposureTime, EXIF FNumber,
get_orientation_from_image(image_path)	Returns the image orientation for the image at the given path from the EXIF metadata.
is_on_path(prog)	Returns true if a certain program is on the environment variable PATH.
set_tesseract_exe()	Sets the path to the executable of tesseract.
remove_filename_duplicates(data_frame)	Removes duplicates entry from a pandas data frame based on the column NAME.

pypocquant.lib.utils.**create_quality_control_images**(*results_folder_path: str*, *basename: str*, *map_of_images: dict*, *extension: str = '.png'*, *quality: int = 100*)

Save the list of requested quality control images.

Parameters

- **results_folder_path** (*str*) – Full path to the folder where to save the quality control images.
- **basename** (*str*) – Common base name for all quality control images.
- **map_of_images** (*dict*) – Dictionary of keys to be appended to the base name with the corresponding image as value.
- **extension** (*str*) – File extension (format). Optional, default is .png.
- **quality** (*int*) – Image compression quality. Optional, default is 100. This is only considered if format is ".jpg".

pypocquant.lib.utils.**get_project_root**() → Path

Returns project root folder.

Returns project_root

Return type Path

pypocquant.lib.utils.**get_data_folder**() → Path

Returns the value of the environment variable DATA_FOLDER or, if not found, the value if *get_project_root()*.

Returns data_folder

Return type Path

pypocquant.lib.utils.**image_format_converter**(*directory*, *filename*, *output_dir=None*, *image_format='tif'*)

Converts a image in raw format (.'nef') to the specified open format. Default is '.tif'.

rawpy API: **https://letmaik.github.io/rawpy/api/rawpy.RawPy.html**, https://letmaik.github.io/rawpy/api/rawpy.Params.html

Parameters

- **directory** – Image directory
- **filename** (*str*) – Filename of the image to be converted
- **output_dir** – Output directory to write the converted image to.
- **image_format** (*str*) – Format of the image such as i.e. tif

pypocquant.lib.utils.**get_iso_date_from_image**(*image_path*)
Returns the date in iso-date format for the image at the given path.

Parameters **image_path** (*str*) – Path to an image.

Returns iso_date

Returns iso_time

pypocquant.lib.utils.**get_exif_details**(*image_path*)
Returns the Exif metadata for the image at the given path. In particular EXIF ExposureTime, EXIF FNumber, EXIF FocalLengthIn35mmFilm, EXIF ISOSpeedRatings.

Parameters **image_path** (*str*) – Path to an image.

Returns exp_time

Returns f_number

Returns focal_length_35_mm

Returns iso_speed

pypocquant.lib.utils.**get_orientation_from_image**(*image_path*)
Returns the image orientation for the image at the given path from the EXIF metadata.

Parameters **image_path** (*str*) – Path to an image.

Returns orientation

pypocquant.lib.utils.**is_on_path**(*prog*)
Returns true if a certain program is on the environment variable PATH.

param prog Name of a program

type prog str

Return type boolean

pypocquant.lib.utils.**set_tesseract_exe**()
Sets the path to the executable of tesseract.

pypocquant.lib.utils.**remove_filename_duplicates**(*data_frame*)
Removes duplicates entry from a pandas data frame based on the column NAME. :param data_frame:

Pandas data frame

Returns data_frame

Return type pd.DataFrame

CHAPTER

EIGHT

PYPOCQUANT - A TOOL TO AUTOMATICALLY QUANTIFY POINT-OF-CARE TESTS FROM IMAGES

This repository contains the implementation of *pyPOCQuant* to automatically detect and quantify test line (TL) signal bands from lateral flow assays (LFA) images, as described in the paper:

- Cuny, A. P., Rudolf, F., & Ponti, A. (2020). pyPOCQuant - A tool to automatically quantify Point-Of-Care Tests from images. MedRxiv,. https://doi.org/10.1101/2020.11.08.20227470

Please cite the paper(s) if you are using this code in your research or work.

8.1 Overview

src/main/resources/base/img/ReadmeFigure-01.png

We developed pyPOCQuant to quantify lateral flow assays (LFA) based Point of Care tests (POCT) from images. The above figure shows an image of a POCT placed on our QR code template as well as a QR code label providing metadata about the sample and test. The POCT gets extracted from the QR code box and finely aligned prior to the detection of the test lines (TLs) from the sensor area. The TLs and their signal strength get quantified after a background subtraction and the results are compiled in a table along with the metadata of the tests automatically for each image.

For a more detailed description please read the user manual or the paper.

8.2 Installation

This package requires Python 3.6 and runs on various platforms. If not explicitly stated differently all the steps below are the same on each platform.

8.2.1 Install | run compiled binaries

The easiest way to run *pyPOCQuant* is to use the compiled binaries which includes everything (except tesseract and zbar, see below) ready to be used.

- download pyPOCQuantUI binaries

8.2.2 Install python and all requirements | run from source

Windows

Install tesseract.

Linux

Install the following dependences (instructions for Ubuntu Linux):

```
$ sudo apt install libzmq3-dev, tesseract-ocr, libzbar0
```

macOS

To install the required dependencies we recommend to use the packaging manager `brew`. Install it from here if you have't allready Install brew.

```
$ brew install zbar
$ brew install tesseract
```

All platforms

pyPOCQuant requires python 3.6. It is recommended to use miniconda: https://docs.conda.io/en/latest/miniconda.html. When miniconda is installed, start the terminal and type:

```
# Create and activate an environment
$ conda create -n pypocquant python=3.6
$ conda activate pypocquant
```

Clone the repo.

```
git clone git://git.gitlab.com/csb.ethz/pypocquantui.git
```

Then, install all requirements.

```
$ cd ${pyPOCQuantUI_root_folder}
$ pip install -r requirements/${platform}
```

where `${platform}` is one of `win32.txt`, `linux.txt`, or `osx.txt`.

Run the GUI with (from within `${pyPOCQuantUI_root_folder}`):

```
$ fbs run
```

For other ways to use *pyPOCQuant* please read the documentation.

8.2.3 Build pyPOCQuantUI

To compile and create a pyPOCQuantUI installer, perform following steps. In the following {ppcqui_root} points to the root folder of the pyPOCQuantUI checked-out code.

Windows

```
$ cd ${ppcqui_root}
$ python ./make_build.py
```

You will find the installer in ${ppcqui_root}\target\pyPOCQuant.

Linux

```
$ sudo apt install ruby ruby-dev rubygems build-essential
$ sudo gem install --no-document fpm
$ cd ${ppcqui_root}
$ python ./make_build.py
```

This will create a ${ppcqui_root}/target/pyPOCQuant/pyPOCQuant.deb package that can be installed and redistributed.

```
sudo apt install ${ppcqui_root}/target/pyPOCQuant/pyPOCQuant.deb
```

Please notice that client machines will need to install also two dependences:

```
sudo apt install tesseract-ocr, libzbar0
sudo apt install ${ppcqui_root}/target/pyPOCQuant/pyPOCQuant.deb
```

8.2.4 macOS

```
$ cd ${ppcqui_root}
$ python ./make_build.py
```

Notes

- Depending on your Python installation, you may need to use pip3 instead of pip.
- For both running it from source or with the compiled binaries zbar and tesseract needs to be installed and be on PATH. On Windows zbar libs are installed automatically.

8.3 Usage

We provide an example workflow in a Jupyter notebook that illustrate how this library can be used as well as a step by step **QuickStart** (add link) guide in the documentation.

8.3.1 Example data

We provide example data as well as an example configuration in this repo under:

```
examples/config.conf
examples/images
```

8.3.2 Creating a config file

In the following we present a brief overview how to create a working config file for your images. Detailed instructions and the definition of each parameter can be found in detail in the manual and documentation. We show how to obtain position and extent of the sensor areas in Fiji or ImageJ. Later we will see how to do the same in the *pyPOCQuant* user interface (GUI).

Important parameters are the `sensor_size`, `sensor_center`, and `sensor_search_area` (the latter being an advanced parameter).

src/main/resources/base/img/strip_annotated.png

Creating a config file with Fiji

1. Open a settings file (i.e default settings) and adjust the parameters to fit your images.
2. Load an image with Fiji and crop it to the size of the POCT

src/main/resources/base/img/fiji_selection.png

1. After drawing a rectangular region of interest, the size is displayed in Fiji's toolbar; e.g. `x=539, y=145, **w=230, h=62**`.
   - When hovering over the central pixels in the top or left sides of the selection, the `x`, and `y` coordinates of the center, respectively, are show in Fiji's toolbar; e.g. `x=*601*, y=144, value=214` (and equivalently for `y`).
2. With the line tool the distance from the border to the test lines (TLs) can be measured and expressed as relative ration (distance to TL from left border / w) to obtain the `peak_expected_relative_location`.

Creating a config file with the GUI

A settings file must not necessarily be created in advance. The Parameter Tree can be edited directly. Optionally, settings can be loaded or saved from the UI.

1. `Select the input folder` and click on one of the listed images to display it. The POCT region will be automatically extracted and shown in the view at the top. The lower view shows the whole image.
2. Hit the `Draw sensor outline` icon (red arrow) in the toolbar. This will allow you to interactively define the `sensor area` and the `peak_expected_relative_location` parameters.

Drawing sensor by clicking into the corners	Drawing finished with aligned test lines (vertical lines)

1. Draw the four corners of the sensor and place the vertical bars on the test lines (TLs). This will cause all relevant parameters to be populated in the Parameter Tree. Please notice that, by default, the `sensor_search_area` is set to be 10 pixels wider and taller than the `sensor_size`. This can be changed in the advanced parameters (but beware to keep it only slightly larger than the `sensor_size`: it is meant only for small refinements).

src/main/resources/base/img/ui_new_settings.JPG

1. Save the settings file (Ctrl+S, `File->Save settings file`) or test current parameters on one image by clicking the `Test parameters` button under the Parameter Tree.

8.3.3 Minimal example

Create a Python script or Jupyter notebook cell with the following code to run the pipeline on all images for a given `input_folder_path`.

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

# Get the default settings
settings = default_settings()

# Change settings manually as needed
settings["sensor_band_names"] = ('igm', 'igg', 'ctl')

# Alternatively, load existing settings file
# from pypocquant.lib.settings import load_settings
# settings = load_settings('full/path/to/settings/file.conf')

# Set final argument
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,
```

(continues on next page)

(continued from previous page)

```
    **settings,
    max_workers=max_workers
)
```

8.3.4 Command line interface (CLI)

Running *pyPOCQuant* from the CLI is best suited when automating the processing of large amounts of images and folders.

To create a default configuration from the CLI, use the `-c` flag of pyPOCQuant.py.

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

By far the easiest approach is to use the *pyPOCQuantUI* (GUI) for this purpose, but it could also be done with other tools, such as Fiji (as described in the manual).

Once the configuration file is ready, a full study can be started by running pyPOCQuant on a full folder of images The analysis is performed in parallel, and the number of concurrent tasks can be adjusted by the `-w` (`--workers`) argument.

```
python pyPOCQuant.py f /PATH/TO/INPUT/FOLDER o /PATH/TO/RESULTS/FOLDER s /PATH/TO/
↪CONFIG/FILE w ${NUMWORKERS}
```

- **-f** `/PATH/TO/INPUT/FOLDER/MANUFACTURER`: path to the folder that contains all images for a given camera and manufacturer.
- **-o** `/PATH/TO/RESULTS/FOLDER`: path where the results (and the quality control images) for a given camera and manufacturer will be saved. The results are saved in a `quantification_data.csv` text file.
- **-s** `/PATH/TO/CONFIG/FILE`: path to the configuration file to be used for this analysis. Note that a configuration file will be needed per manufacturer and (possibly) camera combination.
- **-w** `NUM_WORKERS`: number of parallel processes; e.g. `8`.
- **-v**: `VERSION` : displays current version of *pyPOCQuant*.
- **-h** `HELP`: displays the CLI arguments and their usage.

To run it with the provided example data type:

```
python pyPOCQuant.py f examples/images o examples/images/results s examples/config.
↪conf w 4
```

8.3.5 Graphical user interface (GUI)

We also provide a graphical user interface *pyPOCQuantUI* that enables interactive parameter configuration, parameter testing, and parallel processing of all files in a folder. The UI also offers a graphical tool to create custom sample identifier QR codes, and another to split images by vendor (either by keyword or QR code tag).

Detailed installation and usage instructions can be found in the manual and documentation.

To start the GUI from source navigate into the `pyPOCQuantUI` root folder and run:

```
fbs run
```

or double click on the pyPOCQuant icon installed by the installer or directly on the downloaded binaries.

After selecting the `INPUT FOLDER` and clicking on an image (e.g. `IMG_9068.JPG` in the figure below), the POCT gets extracted and displayed on the right top. Clicking on the `Draw sensor` button (red arrow) allows to identify the sensor area by clicking into its corners. After aligning the relative position of the test lines (TLs) by dragging the vertical lines the button `Test parameters` will open the `OUTPUT FOLDER` and show the results for the selected image. Clicking the button **`Run`** will apply the parameters to all images in the selected folder and process each image in parallel.

src/main/resources/base/img/ui_drawing_arrow.JPG

8.4 Troubleshooting

Installation requires Python 3.6 , PyQT 5 and fbs 0.9 with PyInstaller 3.4. We have tested the package on (macOS, Linux, Windows 7 and 10) Please open an issue if you have problems that are not resolved by our installation guidelines above.

8.5 Contributors

pyPOCQuant is developed by Andreas P. Cuny and Aaron Ponti. If you want to contribute and further develop the project feel free to do so!

8.6 How to cite

```
@article{cuny2020,
  author    = {Andreas P. Cuny and Fabian Rudolf and Aaron Ponti},
  title     = {A tool to automatically quantify Point-Of-Care Tests from images},
  journal   = {MedRxiv},
  year      = {2020},
  doi       = {10.1101/2020.11.08.20227470}
}
```

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex

PYTHON MODULE INDEX

p

INDEX

Symbols

A

B

C

D

E

F

G

I

K

L

M

N

P