Wellcounter Manual v1.1

Claus-Peter Stelzer
Research Institute for Limnology, Mondsee, Universität Innsbruck
Email: claus-peter.stelzer@uibk.ac.at

This manual is a part of the publication:

> *Stelzer, C.P. & Groffman, D. (2025) "Wellcounter: Automated High-Throughput Phenotyping for Aquatic Microinvertebrates". Methods in Ecology and Evolution*

Future versions of this manual will be available at: https://github.com/cpstelzer/wellcounter

Table of Contents

Introduction

The Wellcounter is an advanced platform for automated high-throughput phenotyping of aquatic microinvertebrates (100–2000 µm) using common multiwell plates. It facilitates large-scale ecological experiments by automating image acquisition, processing, and analysis.

The system consists of:

- Hardware: A high-resolution camera, telecentric lens, motorized linear guides, and darkfield illumination.
- Software: Python-based modules for automated data collection and analysis.

This manual provides step-by-step instructions for building the hardware, setting up the software, and using the Wellcounter effectively.

Building instructions

The Wellcounter device (**Fig. 1**) integrates an open-source linear guide system, custom 3D-printed parts, and high-quality optical components for capturing well-resolved images of samples. This manual provides detailed instructions on constructing your own Wellcounter device, covering the essential components and assembly steps. By following these guidelines, you should be able to replicate our prototype, or build a device with similar characteristics.

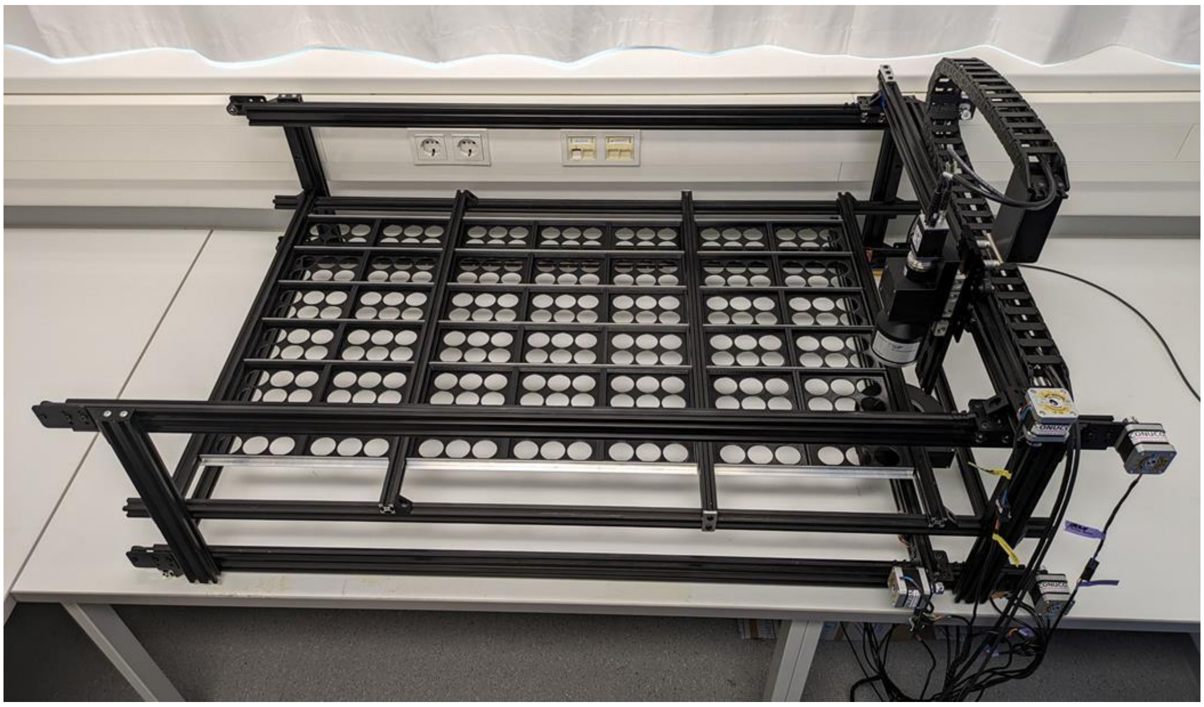


Fig. 1. The Wellcounter device.

Linear guide system

We recommend purchasing a pre-assembled linear guide system to save time and avoid potential mechanical issues from DIY construction. The design we use is based on OpenBuilds (https://us.openbuilds.com), an open-source platform that provides affordable and modular components, including aluminum extrusions, belt drives, and stepper motors. You can buy the components or pre-assembled XY linear guides from OpenBuilds (builds.openbuilds.com), or from a reseller (e.g., conucon.de). Usually, such linear guides come at default dimensions, e.g., 1000 x1000 mm, and consist of a frame with two axes travelling in the X- and Y- direction.

In our Wellcounter design, we customized several aspects:

- We use two XY linear guides that are assembled on top of each other. The linear guide at the upper level holds the camera/lens while one at the bottom level holds the ring illumination (**Fig. 2**).
- The object plane, which holds the six-well plates, is located in between the two linear guides. It is a fixed rectangular frame consisting of 20x20 mm aluminum extrusions and T-profiles (**Fig. 2 B-E**).
- All three levels are screwed together by four vertical braces consisting of 20x40 mm extrusions. The distance between the top plane and the object plane in our device is 140 mm (**Fig. 2 A, E**).
- Our linear guides have custom dimensions of 750 x 1150 mm, to better fit on the table in our lab

Checklist for ordering a linear guide system:

- Pre-assembled XY linear guide system
- Stepper motors (NEMA 17 or NEMA 23), motor drivers, and cables
- 12V, 5A power supply
- Microcontroller (e.g., OpenBuilds BlackBox, or Arduino with GRBL firmware)

Apart from these components, you will also need a PC, which is connected via USB to the microcontroller. This PC controls the linear guide either through a low-level table control software provided from the manufacturer (e.g., OpenBuilds CONTROL or CONUCON software), or programmatically via Python commands (e.g., in the Wellcounter *acquisition module*). For more details, see the next section (Basic software control).

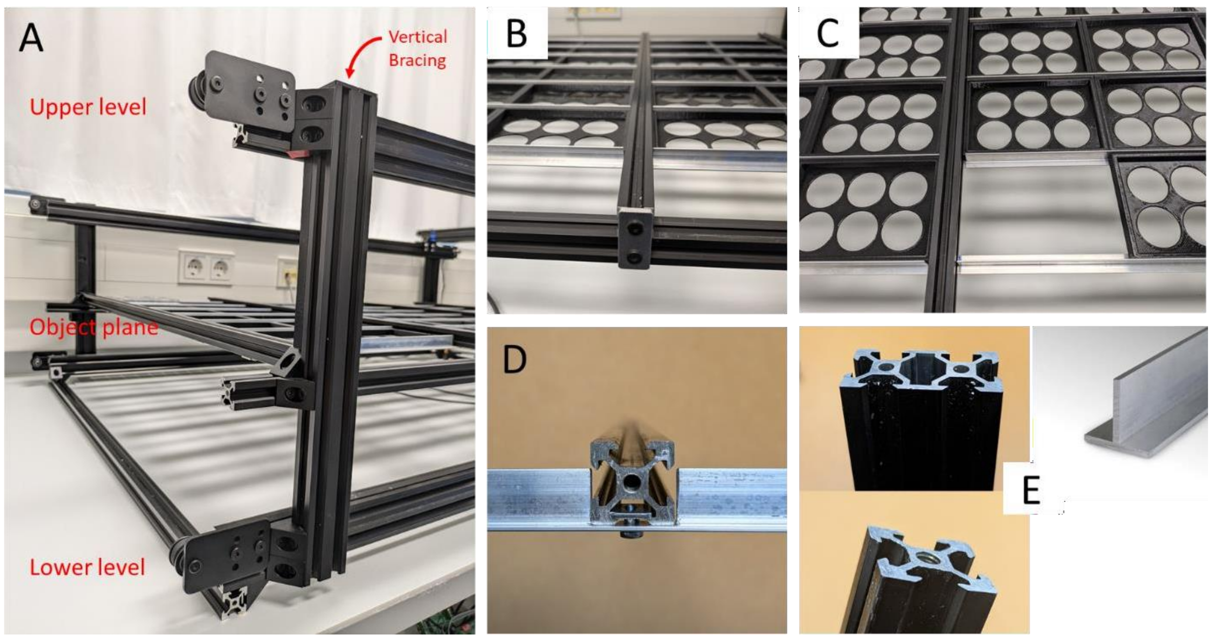


Fig. 2. Components of the Wellcounter device. **A** The three levels (Upper level, object plane, lower level) of the Wellcounter device are held together by four vertical braces consisting of 20x40mm aluminum extrusions. **B**, **C** The object plane consists of 20x20mm extrusions and aluminum T-profiles, which hold the black 3D-printed wellplate holders. **D** Connection between T-profiles and 20x20 extrusion. **E** Raw components of the object plane: 20x40 mm and, 20x20mm extrusions, T-profile.

Optical components and illumination

The most important aspect of choosing a camera and lens is ensuring that their optical characteristics (e.g., field of view, working distance) align with the spatial constraints of the objects to be imaged, such as the size of a well or the available distance between the camera/lens and the object plane. Ideally, the optical system should be configured so that one well is displayed in a format that fills the camera's sensor. Online calculators (e.g., baslerweb.com/en/tools/lens-selector/) can assist with selecting suitable optical components.

Our Wellcounter prototype utilizes a monochrome digital camera (Basler a2A4504-27g5m) with a Sony IMX541 CMOS sensor, capable of delivering up to 27 frames per second at a resolution of 20.2 MP (4512 x 4512 pixels). We use a telecentric lens (model: OPT-10M035-110) with 0.35X magnification and a working distance of 110 mm. This optical configuration produces a field of view with a 46.3 mm diameter, precisely covering one well of a 6-well plate (Greiner Bio One, Model: 657160). The camera is connected to a PC through a 5-GigE interface card. Darkfield background illumination is provided by an MBJ horizontal ringlight HDF-07-BE-s (MBJ imaging, Hamburg, Germany) with a peak wavelength of 465 nm (blue light).

To aid in selecting components for similar applications, we offer the following recommendations:

- **Use an industrial-grade camera that can be controlled via Python commands** (e.g., through Basler Pylon). Consumer digital cameras are generally unsuitable, as they lack low-level control capabilities and store images/movies on the camera itself rather than on a computer, where the analysis is performed.
- **If color is not essential to your application, opt for a monochrome camera** because it provides higher sensitivity and greater resolution by capturing more light at each pixel compared to a color sensor. Monochrome cameras avoid the color filter array used in color sensors, allowing for sharper imaging, especially beneficial in low-light conditions or when using specific wavelengths for illumination.
- **Invest in a telecentric lens.** Although telecentric lenses are considerably more expensive than standard lenses, they produce an essentially distortion-free image, ensuring accurate representation of each object's position and size.
- **Ensure that no downstream component limits the camera's frame rate.** Use a suitable high-bandwidth interface card for the computer if needed. For example, we added a 5-GigE interface card to our setup.
- **Monochrome illumination, ideally at short wavelengths, offers superior sharpness.** We use blue light illumination at 465 nm, but other wavelengths can also be effective.

3D-printed components

In the Wellcounter, we use 3D-printed adapters to mount the camera/lens and ring light onto the linear guides, or as holders for 6-well plates onto the object plane (**Fig. 3**). We used black PETG as printing material, which is cost-efficient, easy to print, and durable.

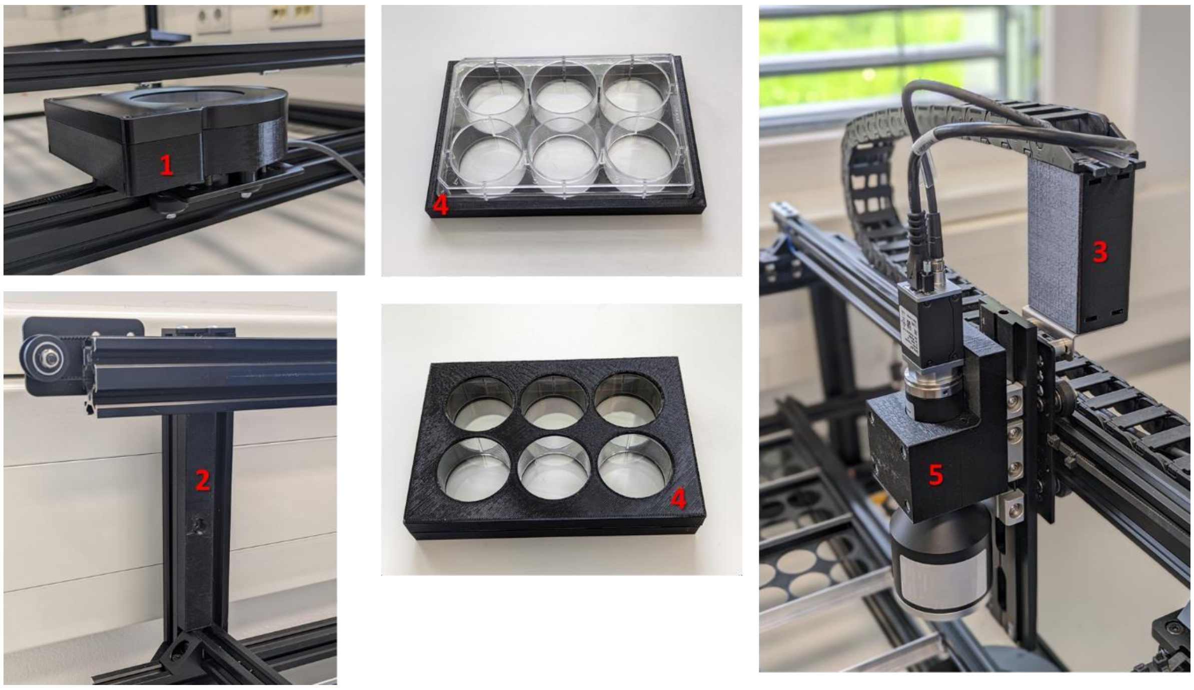


Fig. 3. 3D-printed parts of the Wellcounter. **1** Mounting adapter for ring illumination, **2** Alignment rail for adjusting the distance between upper level and object plane, **3** Spacer for drag chain, **4** Well plate holder used to mount well plates onto the object plane, or can be placed on top of a well plate to block light signals outside the well, **5** Mounting adapter for camera and lens.

If you do not have access to a 3D-printer, you can upload the provided STL-files to a 3D printing service. However, if you chose this approach, we recommend having a test print done for one of the 6-well plate holders before printing the full set of 84 pieces, since the tolerances of 3D printed parts may vary from printer to printer.

The 3D printed parts are specifically designed for the optical components stated above. They will probably not fit for different models, or if manufacturers chose to change the dimensions of the existing models. In these cases, it would be necessary to design new adapters that match the dimensions of the new optical components.

Basic software control

The goal of this section is to familiarize you with software control of the Wellcounter device. We will install the Conda environment, which contains all the dependencies for running the Wellcounter software written in Python. We will also generate a csv fie in which the coordinates to the physical positions of all plates and wells on the object plane are stored.

Background

At the low level, the XY linear guide system of the Wellcounter is controlled by commands written in G-code, which tell the motors how to move. This G-code is sent to the microcontroller, which converts it into electrical signals that ultimately drive the stepper motors translate it into actions like moving along the X and Y axes.

Example of a G-code command:

```
G1 X5 Y10
```

This moves the device 5 units in the X-axis and 10 units in the Y-axis. Your linear guide system should come with a software that allows directly entering G-code commands.

G-code commands can also be sent to the microcontroller from within a Python-program using the package pyserial, which is the method how the Wellcounter interfaces with the linear guide. Here is the same command send from within Python:

```
gcode_command = 'G1 X5 Y10\n'

ser.write(gcode_command.encode('utf-8'))
```

Note that '\n' (newline) indicates the end of the command.

Setting up the Conda environment

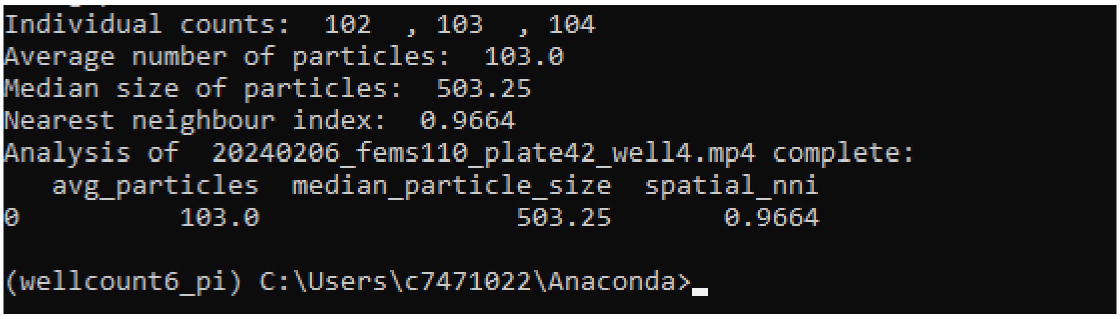
To run the Wellcounter software on your local computer, you need to first replicate the required environment with all its dependencies. The easiest way to accomplish this is to install the package manager Anaconda on your computer (http://anaconda.org/). After installing Anaconda, complete the following steps:

1) Copy the Wellcounter environment-file to a destination folder on your PC, e.g.:
   ```
   C:/Users/Anaconda/wellcount6_pi.yml
   ```
2) Install the environment on your PC with the following command:
   ```
   conda env create -f C:/Users/Anaconda/wellcount6_pi.yml
   ```
   This step only needs to be done once.
3) After the environment has been installed, open the Anaconda terminal and activate the Wellcounter environment by typing:
   ```
   conda activate wellcount6_pi
   ```
   The terminal should now display "(wellcount6_pi)" followed by your current working directory.
4) Copy all the Wellcounter software files into your working directory. This step only needs to be done once.

You can verify the correct installation of your Wellcounter Conda-environment by running an analysis of pre-recorded sample movies. To run one of the example scripts, e.g., *wc_analyze_one_sample.py*, follow these steps:

- Copy the mp4-movie file that is to be analyzed into any directory of your choice.

- Open the example script *wc_analyze_one_sample.py* in an editor of your choice (e.g., Spyder, which is part of the Anaconda package) and adjust the variables 'data_dir' and 'video_file' to match the location of your video-file. Save these changes.
- To execute the script, go to the Anaconda terminal ('wellcount6_pi' activated), navigate to the directory where your Wellcounter software is stored, and and type:
  `python wc_analyze_one_sample.py`
- Depending on the hardware of your computer, the analysis will take 20-60 seconds (if only counting is done), or a few minutes (if motion analysis is done as well) and it will output the main results on the screen (**Fig. 4**)

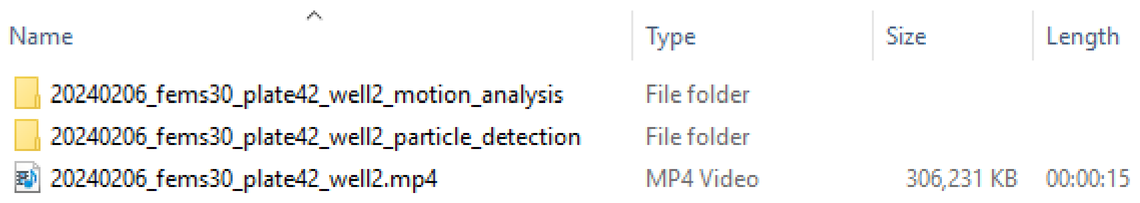


```
Individual counts:  102  , 103  , 104
Average number of particles:  103.0
Median size of particles:  503.25
Nearest neighbour index:  0.9664
Analysis of  20240206_fems110_plate42_well4.mp4 complete:
   avg_particles  median_particle_size  spatial_nni
0          103.0                503.25       0.9664

(wellcount6_pi) C:\Users\c7471022\Anaconda>_
```

Fig. 4. Typical screen output after running image analysis.

Furthermore, after the analysis is complete, you will find two new folders in the same directory as your mp4-file, which contain the results of particle detection and motion analysis, respectively (**Fig. 5**).



Name	Type	Size	Length
20240206_fems30_plate42_well2_motion_analysis	File folder		
20240206_fems30_plate42_well2_particle_detection	File folder		
20240206_fems30_plate42_well2.mp4	MP4 Video	306,231 KB	00:00:15

Fig. 5 Folders containing the results of image and motion analysis.

Generating the *wellpositions_all.csv* file

During recording of an experiment, the Wellcounter moves the camera/illumination unit to each individual well in the object plane, thus visiting a total of up to 42 x 6 = 252 wells. The G-code commands required for this movement are calculated based on the positions of each of these 252 wells in the object plane. Before the first use of the Wellcounter, the exact positions of these wells have to be empirically determined and stored in a file called wellpositions_all.csv. This procedure needs to be done only once, unless you make changes in the object plane that alter the physical arrangement of the well plates. As a reference point (X0, Y0) of the coordinate system of the object plane, we use the center of "well 6" of "plate 42", which is located in the low right corner of the object plane (Fig. 7). The workflow for generating the wellpositions_all.csv file is as follows:

- Put a small piece of white cardboard behind the ring illumination, to create a white background (see below, **Initial calibration** in Workflow1: Data Acquisition). Place all 42 wellplate holders on the object plane.
- Start your low-level table control software provided with the XY linear guide (e.g., CONUCON software) and the camera control software of your camera manufacturer (e.g., Basler Pylon).
- Move the camera and illumination units to "well 6" of "plate 42". Tare this position to the coordinates 0, 0 (X, Y). This procedure is described in more detail below (see below, **Initial calibration** in Workflow1: Data Acquisition).
- Now move the camera and illumination to "well 6" of "plate 41". Note the XY-coordinates of well 6 of plate 41 and continue with each the remaining 40 plate holders.
- Enter all 42 coordinates in a text editor and save the results as a csv-file (wellpositions_w6.csv), which will look like this

```
plate,well,originX,originY
1,6,86.680,48.448
2,6,86.714,38.704
3,6,86.740,28.986
4,6,86.775,19.256
…
```

- To calculate the positions of the remaining wells no. 1-5 of each plate, you can use our Python script calculate_all_wellpositions.py. This script outputs the file wellpositions_all.csv which contains the coordinates of all 42x6=252 wells:

```
plate,well,X,Y
1,1,94.48,52.35
1,2,90.58,52.35
1,3,86.68,52.35
1,4,94.48,48.45
1,5,90.58,48.45
1,6,86.68,48.45
2,1,94.51,42.6
…
```

Using the Wellcounter

Before you start using the Wellcounter, make sure you have all the hardware set up, all hardware drivers installed (e.g., linear guide, camera), and the 'wellcount6_pi' Conda-environment installed and activated. In the following, we describe three workflows, data acquisition, image and motion analysis, and optimize imaging parameters.

Workflow1: Data Acquisition

This workflow guides you through capturing experimental data using the Wellcounter acquisition module. The purpose of the acquisition module is to automate the recording of experimental data by controlling hardware components to capture images and videos of samples in multiwell plates. The Wellcounter can be loaded with up to 42 six-well plates (6 rows x 7 columns, **Fig. 6**). The camera/light-unit can travel to each well and take photos or record movies.

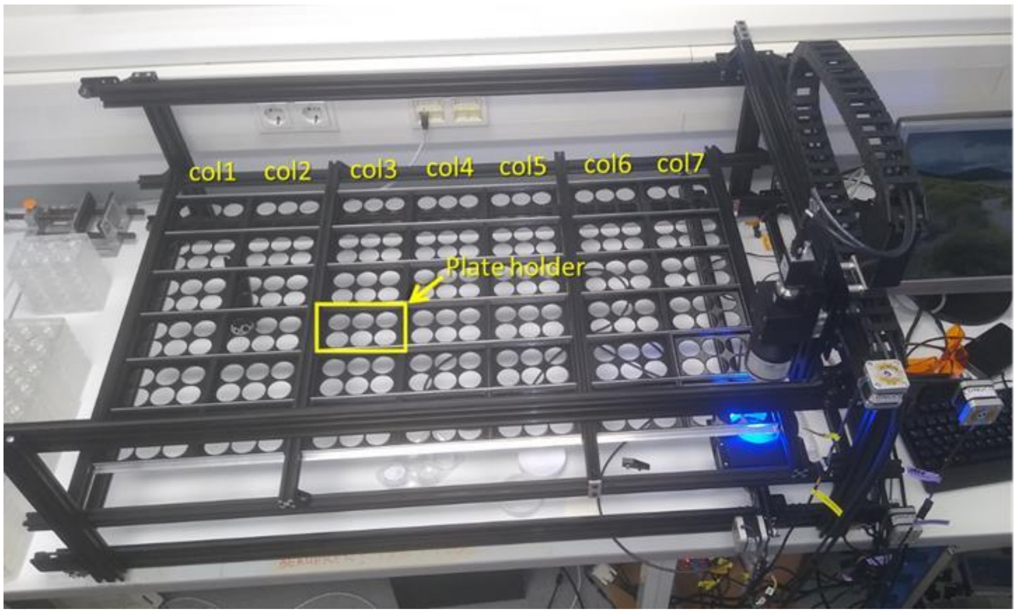


Fig. 6. Six-well plate holder arrangement.

The camera/light-unit travels to the different plate and well positions according to the paths indicated in light blue, from plate 1 to plate 42 (**Fig. 7**). The origin of the entire coordinate system is plate 42, well 6 (Position: X0 Y0) while the most extreme position is plate 1, well 1 (In our model, position: X94.48 Y52.35).

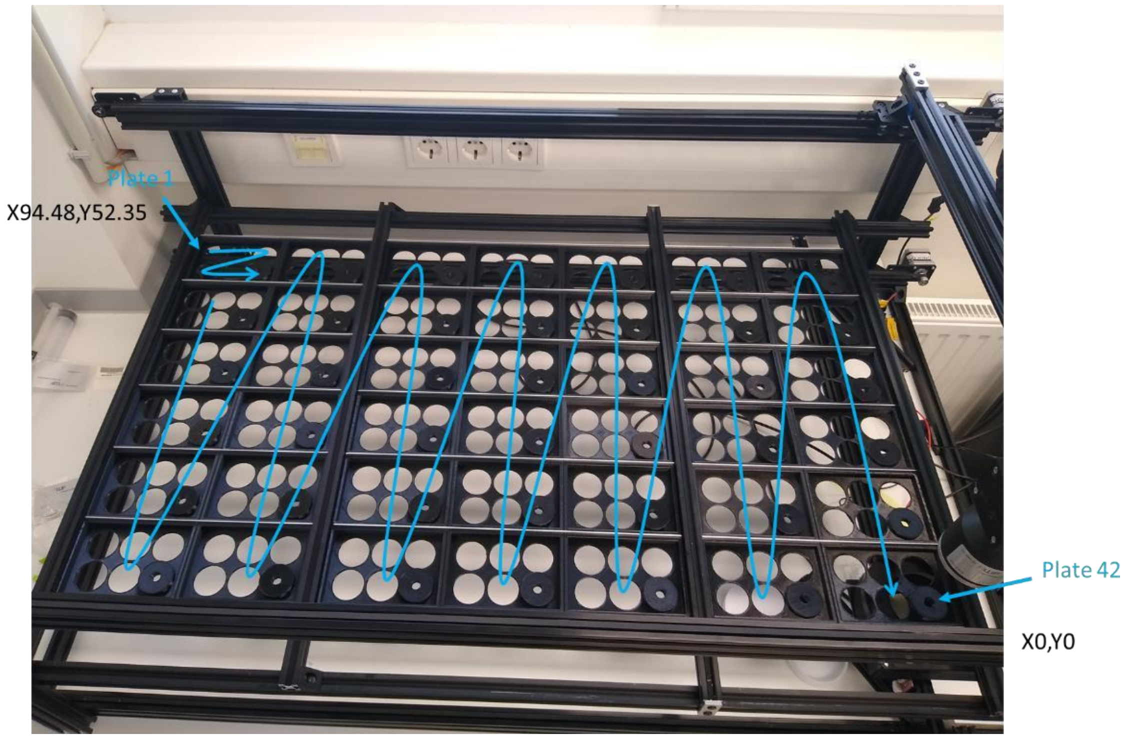


Fig. 7. Traveling paths of the camera and illumination unit.

Check Hardware Connections:

- Connect the digital camera, XY-scanning table, and USB relay to your computer.

- Verify the serial ports used by the devices and update the script accordingly (e.g., ser.port = 'COM5' for the XY-table, portName = 'COM4' for the USB relay).

Configure Paths and Parameters:

- Open wc_record_experiment.py in an editor.
- Set the *output_folder* and *movies_folder* variables in the script to specify where images and videos will be saved.
- Adjust movement parameters such as speed and acceleration if needed.
- Ensure the *csv_file* variable points to the correct CSV file containing well positions.

Prepare the wellpositions CSV file:

- Create or verify the CSV file (e.g., wellpositions_all.csv) containing the X and Y coordinates of each well. See above, Generating the *wellpositions_all.csv* file.
- The CSV should have columns like plate, well, originX, originY.

Initial calibration

Before each run, a calibration of the XY linear guide is recommended. In this procedure, you assign the coordinates X0, Y0 to the 6th well of the 42nd plate (see Fig. 5). Below, we describe our protocol using the CONUCON software and Basler Pylon software (for low-level table and camera control, respectively):

1. By default, the background of the ring illumination is black (**Fig. 8**: 1). Place a white piece of paper behind the background illumination to create a white background (2). Put back the plate holder (3).

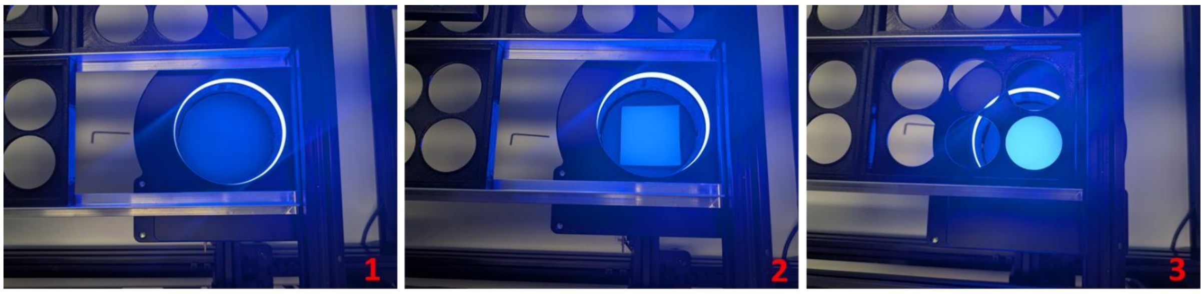


Fig. 8. Creating a white background for calibration of wellpositions.

2. *Pylon Viewer*: Select the camera (**Fig. 9:** 1), switch it on (2), and turn on “live view” (3). Check the image: You should see a white circle that is perfectly centered. If this is not the case, you have to align the camera as outlined in the next step with the CONUCON software.

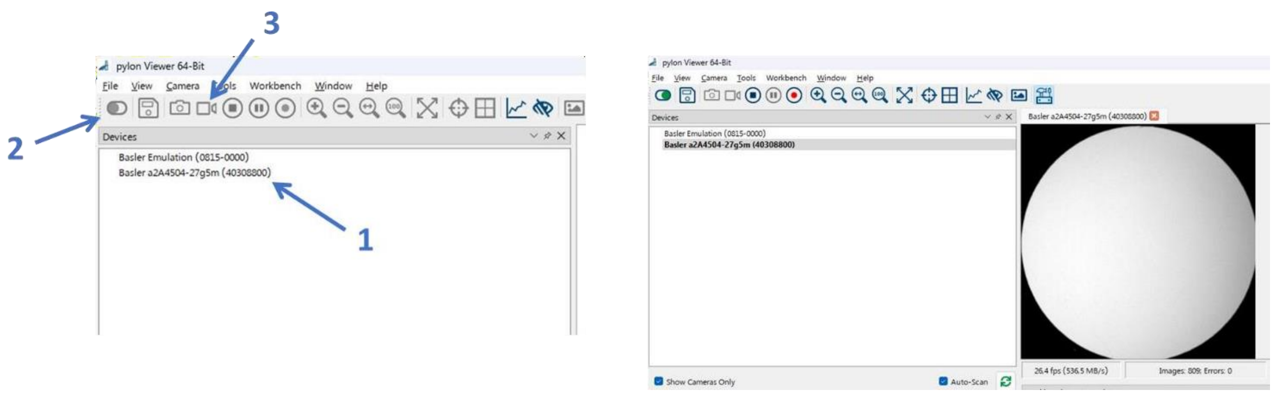
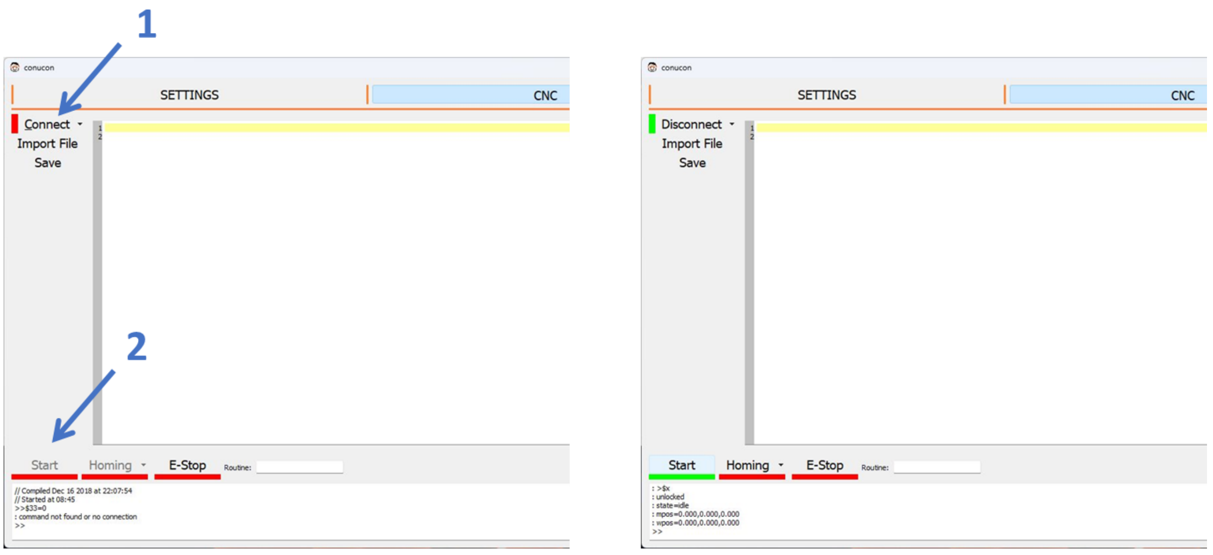

3
pylon Viewer 64-Bit
File View Camera Tools Workbench Window Help
2
Devices
Basler Emulation (0815-0000)
Basler a2A4504-27g5m (40308800)
1
Basler a2A4504-27g5m (40308800)
Show Cameras Only
Auto-Scan
26.4 fps (536.5 MB/s)
Images: 809; Errors: 0

Fig. 9. Calibration procedure: Establishing connection to camera.

3. CONUCON software: Press “Connect” (Fig. 1) and then press “Unlock” (2). The two Buttons should then light on as green and display “Disconnect” and “Start”, respectively. To align the camera, enter updated X or Y positions into the command line of the CONUCON software.



```
// Compiled Dec 16 2018 at 22:07:54
// Started at 08:45
>>$33=0
: command not found or no connection
>>
```

```
: >$x
: unlocked
: state=idle
: mpos=0.000,0.000,0.000
: wpos=0.000,0.000,0.000
>>
```

Fig. 10. Calibration procedure: Establishing connection to XY linear guide.

4. Enter the values as numbers preceded by X or Y, e.g.: `X-0.05`. When you press return, the table should move. <u>The values you enter here are positions, not movements!</u> Make sure you only enter small values for the refined position (e.g., 0.05 – 0.1).

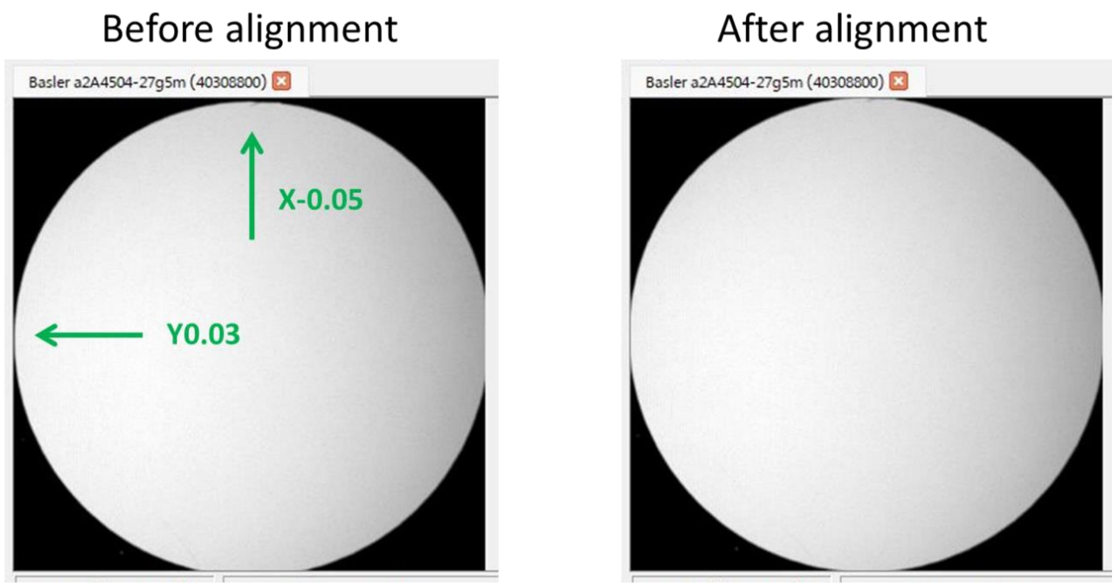
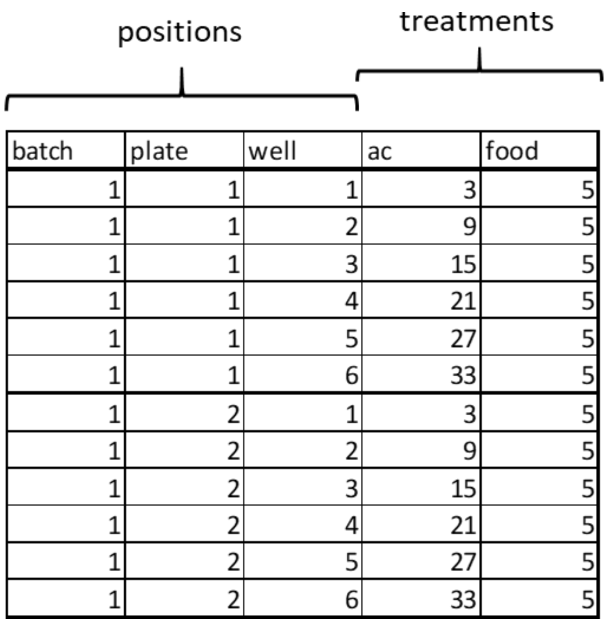

Before alignment
After alignment
Basler a2A4504-27g5m (40308800)
Basler a2A4504-27g5m (40308800)
X-0.05
Y0.03

Fig. 11. Centering the reference well at X0 Y0.

5. Once the white circle is perfectly centered (**Fig. 11**), press “Disconnect” and close the CONUCON program. This effectively tares the current position to X0 Y0. Then, switch off the camera and close the Pylon Viewer program. It is essential to end both programs, otherwise it will cause conflicts with the measurement program.
6. Remove the white cardboard piece from the ring lamp and place back plate holder 42.
7. The Wellcounter is now ready for image acquisition!

Prepare the Experiment:

- Place all your six-well plates with the microorganisms (in 5ml culture medium) on the XY-scanning table according to your experimental design.
- If you want to make full use of the Wellcounter’s high throughput capabilities you should prepare an Excel file containing information about which treatment is linked to which batch/well/plate, from which you export a treatments.csv file (**Fig. 12**). You can use this treatments.csv file later for automatically analyzing an entire experiment with thousands of recorded movies.


positions
treatments

batch	plate	well	ac	food
1	1	1	3	5
1	1	2	9	5
1	1	3	15	5
1	1	4	21	5
1	1	5	27	5
1	1	6	33	5
1	2	1	3	5
1	2	2	9	5
1	2	3	15	5
1	2	4	21	5
1	2	5	27	5
1	2	6	33	5

Fig. 12. Example of storing experimental information in the treatments.csv file.

- Remove lids and place a lid covers (**Fig. 13**) on each plate, and ensure the room is darkened.

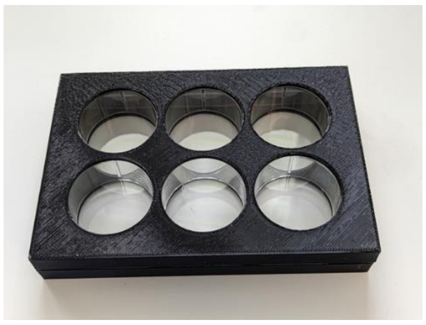
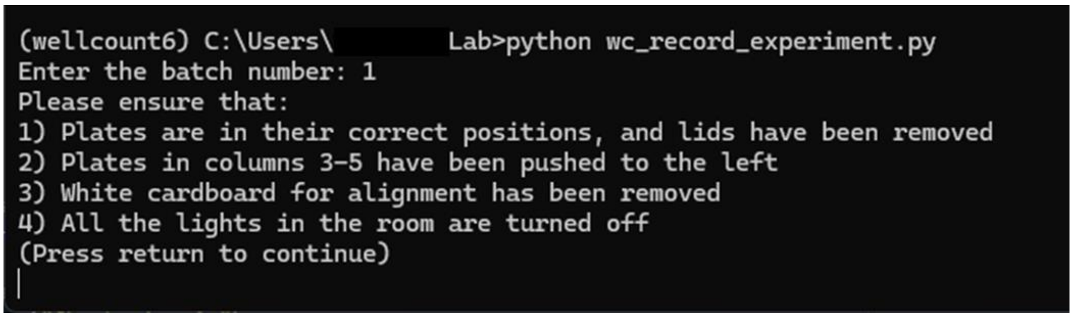


Fig. 13. Wellplate with lid cover on the top.

Run the script:

- Activate the Wellcounter Conda-environment
- Execute wc_record_experiment.py in your Python environment.
- When prompted (**Fig. 14**), enter the batch number (an integer representing your experiment batch).
- Check and confirm that all setup conditions are met by pressing Enter.



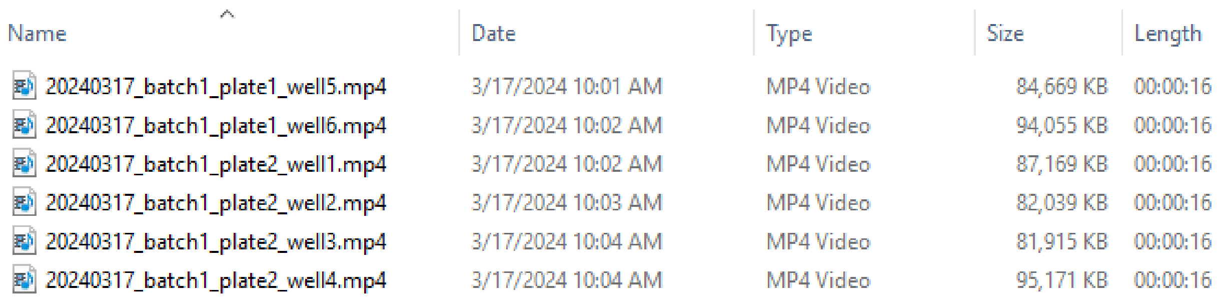
```
(wellcount6) C:\Users\[redacted]Lab>python wc_record_experiment.py
Enter the batch number: 1
Please ensure that:
1) Plates are in their correct positions, and lids have been removed
2) Plates in columns 3-5 have been pushed to the left
3) White cardboard for alignment has been removed
4) All the lights in the room are turned off
(Press return to continue)
|
```

Fig. 14. Initial dialog when running wc_record_experiment.py

Upon running the script, the Wellcounter will move the XY-scanning table to each well position, control the lighting by turning the relay on and off, record videos at each well position, log video recording times and positions.

The output consists of:

- Videos of each well saved in the specified movies_folder. The naming of the video files follows a strict pattern of date, batch number, plate number, and well number, which facilitates automated analysis later on.



Name	Date	Type	Size	Length
20240317_batch1_plate1_well5.mp4	3/17/2024 10:01 AM	MP4 Video	84,669 KB	00:00:16
20240317_batch1_plate1_well6.mp4	3/17/2024 10:02 AM	MP4 Video	94,055 KB	00:00:16
20240317_batch1_plate2_well1.mp4	3/17/2024 10:02 AM	MP4 Video	87,169 KB	00:00:16
20240317_batch1_plate2_well2.mp4	3/17/2024 10:03 AM	MP4 Video	82,039 KB	00:00:16
20240317_batch1_plate2_well3.mp4	3/17/2024 10:04 AM	MP4 Video	81,915 KB	00:00:16
20240317_batch1_plate2_well4.mp4	3/17/2024 10:04 AM	MP4 Video	95,171 KB	00:00:16

Fig. 15. Example of movie files recorded with the Wellcounter.

- A log file (video_log.csv) of the entire experiment, containing additional information, such as frame rate, total frames recorded of positions and recording duration of each recorded movie.

Workflow 2: Image and motion analysis

In the following, we process recorded videos using the *imaging* and *motion module*. Here, we assume that you have already fine-tuned the detection parameters for your focal animals. If you use an organism of similar size or larger than *Brachionus plicatilis* (~200-400µm in length), the default parameters should work. If not, consider the following section (Workflow 3: Optimize detection parameters). If you just want to test whether the Wellcounter software runs on your computer, feel free to use one of our pre-recorded mp4-movies provided as supplementary data with the publication.

Configure Imaging Parameters:

- Open the wellcounter_config.yml file using a code editor of your choice (e.g., MS Visual Studio).
- Check and/or adjust parameters under particle_detection:
    - *microorganism_threshold*: Pixel intensity threshold for detecting particles (e.g., 24).
    - *min_microorganism_area*: Minimum area in pixels for detected particles (e.g., 210).
- Set output options to specify which outputs to generate (e.g., *particle_detection*: `True` to save labeled images).

Process Videos:

- Open an Anaconda terminal, activate your Wellcounter environment (conda activate, and navigate to the folder where your Wellcounter software (modules, scripts) is stored.
- For analyzing one sample, you can use the script wc_analyze_one_sample.py. In this case, you need to provide within the script:
    - The path to the directory of your video file (variable: *data_dir*).
    - The name of your video file (variable: *video_file*).
    - Open the script in a code editor of your choice (e.g., MS Visual Studio), edit these data, and save.
    - Start the analysis by entering `python wc_analyze_one_sample.py` in the Anaconda terminal.
- For analyzing an entire experiment consisting of thousands of videos, you can use the script wc_analyze_experiment.py. In this case, you need to provide within the script:
    - The path to your video files (variable: *data_dir*)
    - The treatments.csv file with information about which treatment is linked to which batch/well/plate (variable: *treat_file*)
    - A name for the csv-file where the results are stored (variable: *outfile*)
    - The start and end date of your experiment as a number in the format 'yyyymmdd' (variables: *start_date* and *end_date*)
    - Open the script in a code editor of your choice (e.g., MS Visual Studio), edit these data, and save.
    - Start the analysis by entering `python wc_analyze_experiment.py` in the Anaconda terminal.
    - The script then iterates through each sample for each day of the experiment, performs analyses on the populations (including counting and motion analysis), and combines the results with the information on the treatments, which are saved in a csv-file specified by the variable *outfile*.

Workflow 3: Optimize detection parameters

To ensure that the Wellcounter reliably and exclusively detects your study organism, it is necessary to specify suitable values for the imaging parameters *microorganism_treshold* and *min_microorganism_area* in wellcounter_config.yml. There are two approaches to do this, (1) by interactively testing a few parameter combinations and (2) using a grid search to examine a wide parameter space and find the best parameter combination.

Interactive approach

In this approach you are changing the parameters manually and then examine their results after running the *imaging module*:

- Open wellcounter_config.yml in a code editor of your choice (e.g., MS Visual Studio).
- Adjust the parameters under particle_detection:
    - *microorganism_threshold*: Pixel intensity threshold for detecting particles (e.g., 24).
    - *min_microorganism_area*: Minimum area in pixels for detected particles (e.g., 210).
- Set outputs options to generate labelled output images (e.g., *particle_detection*: `True` to save labeled images).
- Save and close wellcounter_config.yml
- Run the script wc_analyze_one_sample.py on a video of your choice (see **Process Videos**)
- Examine the labelled output images. If many animals were missed, you may want to decrease the values of *microorganism_threshold* and/or *min_microorganism_area*. If there are many false positives, you should increase the values of *microorganism_threshold* and/or *min_microorganism_area* in the next iteration.

This approach should lead to a reasonably good detection rate after a few iterations. However, it is unlikely to find the optimal parameter combination.

Grid search

A more comprehensive, but also more time-consuming approach to determine the optimal imaging parameter combinations.

Overview of Steps (see **Fig. 16**)

1) Running the *imaging module* on multiple movies with the default parameters (or parameters determined by the interactive approach)
2) Manually classifying false positives using the *review module*
3) Manually classifying false negatives using an image editor (e.g., Irfanview)
4) Compiling a 'ground truth' dataset for all movies (by removing false positives and adding false negatives to the particle data for each movie)
5) Running the optimization module to find the best imaging parameter combination

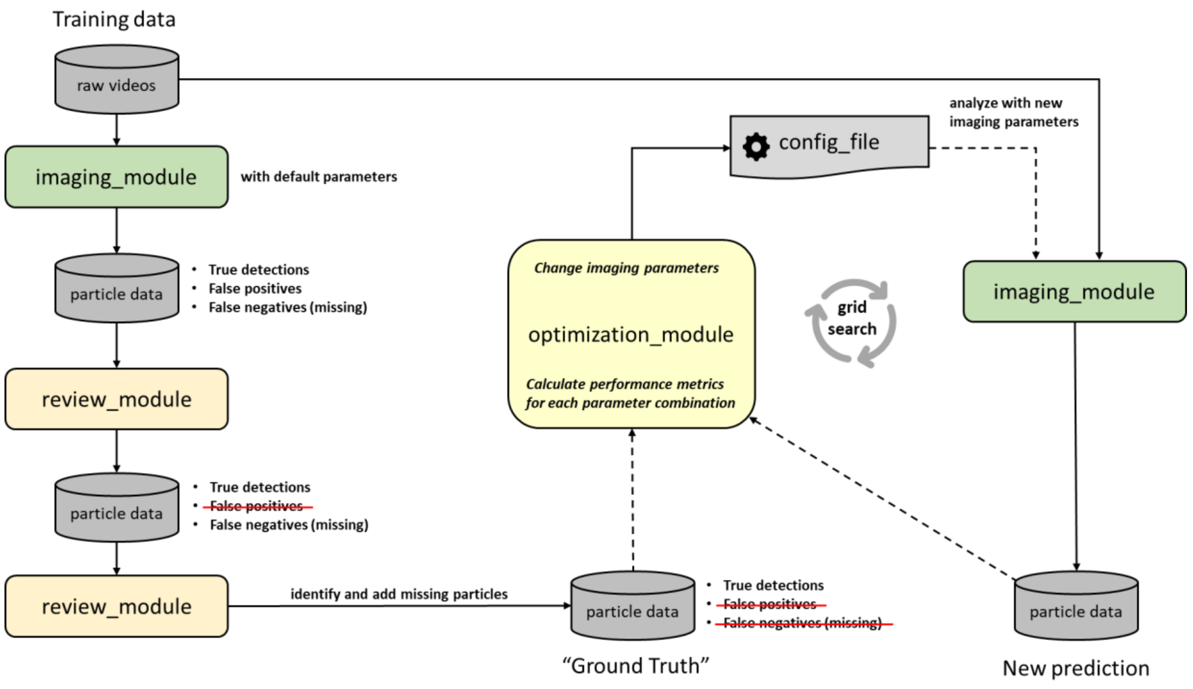


Fig. 16. Schematic of grid search to determine optimal imaging parameters for a new model organism.

This workflow focuses on improving detection accuracy by manually reviewing detected particles and optimizing imaging parameters. It is used when adapting the Wellcounter to a new study organism for the first time, or if major changes in the optical system have been done (e.g., installing a new camera).

Prerequisites

- A set of videos for generating labeled training data.
- The default setting of the imaging parameters *microorganism_treshold* and *min_microorganism_area* should already detect some of your study organisms. You can check this by running wc_analyze_one_samlpe.py on one of your videos and examining the results. If detection rate is very low, or if false positive detections are very high, you should first use the Interactive approach(see above) for finding better initial values for the imaging parameters

1) Run the imaging module with default/initial parameters

- Check if your wellcounter_config.yml file contains the desired default values for *microorganism_treshold* and *min_microorganism_area*
- Comment out the line "motion_df = wmm.perform_motion_analysis(video_path)" in wc_analyze_one_sample.py by entering a # in the beginning of this line. This will suppress motion analysis, which is not useful at the moment.

- Run the image analysis for each movie. This will generate a results folder containing the name of your movie with "_particle_detection" added, as well as a file called table_of_particles.csv.

2) Manually classifying false positives using the review module

- Use the provided script wc_assign_particletypes.py. Before running the script, open it in an editor and read the description on how to use it.
- Assign categories (e.g., true_swimming, true_stationary, false positive) to each particle (**Fig. 17**).

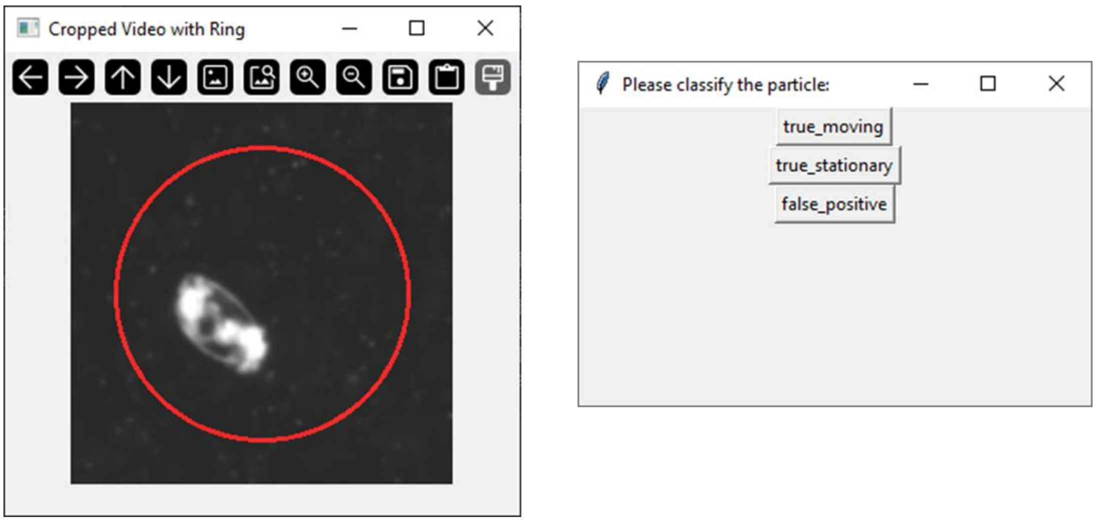


Fig. 17. Example screenshot of wc_assign_particletypes.py (review module).

- Tip: You don't need to watch each movie of each particle to the end. Once you recognize the category, press "q" to skip the rest of the movie and enter the category.
- Output: Running this part of the script will edit your table_of_particles.csv by adding a new column containing your assigned categories, which will be saved as table_of_particles_fp.csv in the same folder.

3) Manually classifying false negatives using the review module

- After all particles have been assigned the categories 'true' or 'false positive', the script wc_assign_particletypes.py will directly continue with the detection of false negatives (i.e., animals that are visible in the first frame, but were not detected by the initial parameter settings).
- To assist in detecting false negatives, the script wc_assign_particletypes.py splits the first frame of the movie into four quadrants, for better visibility. For each quadrant (**Fig. 18**), it displays the annotated image of the first frame (with true positives marked in green and false positives in red) and then repeatedly plays selected frames from later in the video (25. 50, 75% of the video, and the last frame) to help you in spotting false negatives. For each quadrant, there are two phases. In the 'display phase', the program displays the current quadrant and quickly iterates through the different frames of the movie. Once you press "q", you will enter the 'marking phase', where only the first frame is displayed. To record the positions of false negatives, click on all false negative particles that you have identified previously during the animation phase. After all false negatives have been marked, press 'Space' to continue to the next quadrant.

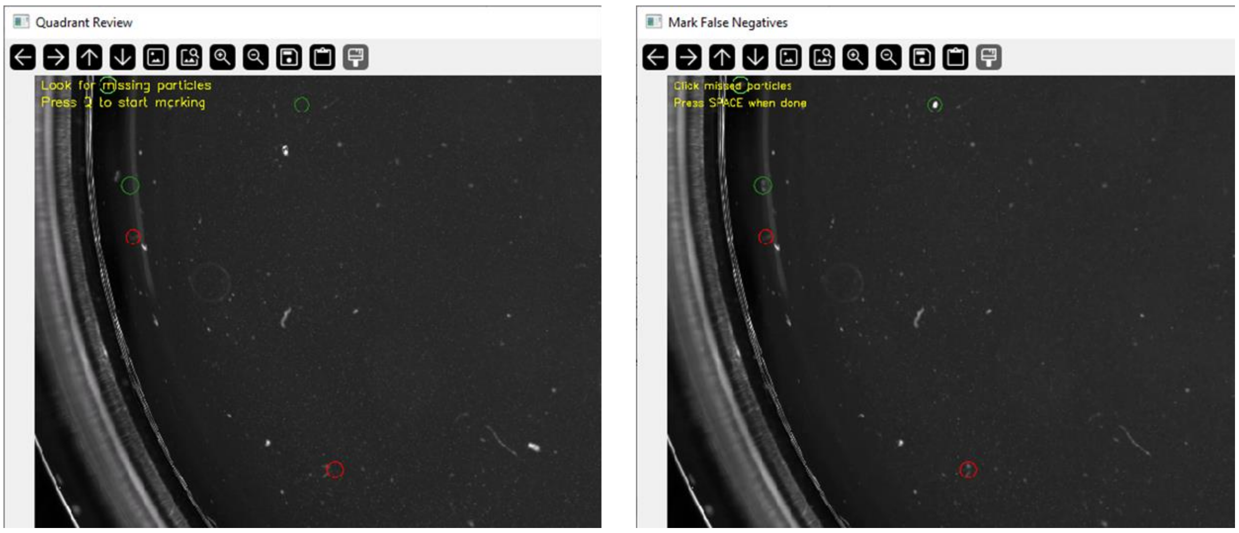


Fig. 18. Example screenshot of wc_assign_particletypes.py (review module) during detection of false negatives. Left: Animation phase. Right: Marking phase. Note that this image does not display the full quadrant.

- After all the four quadrants have been processed, the program will update the table_of_particles variable by adding the coordinates of the false negatives and their particle categorie (value: 4) and save it as table_of_particles_cat.csv. You will also find a new image (first_frame_typelabels.jpg, **Fig. 19**) containing all particles labelled in different colors according to their assigned category (true, false positive, false negative).

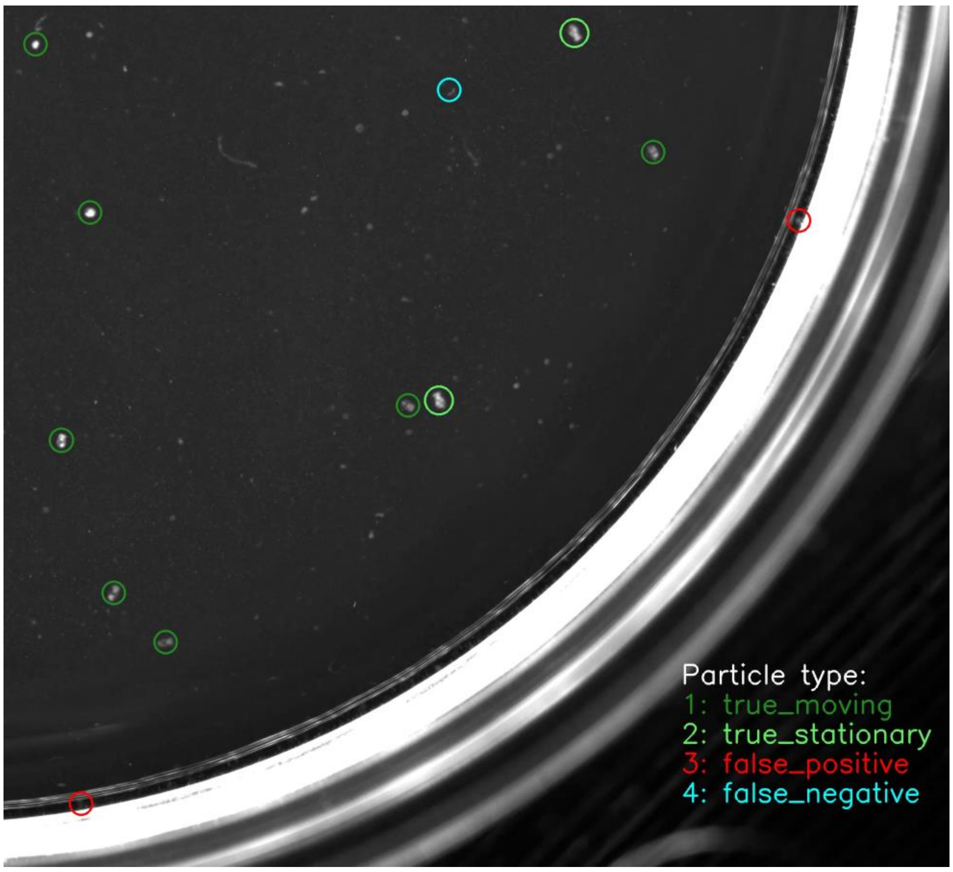


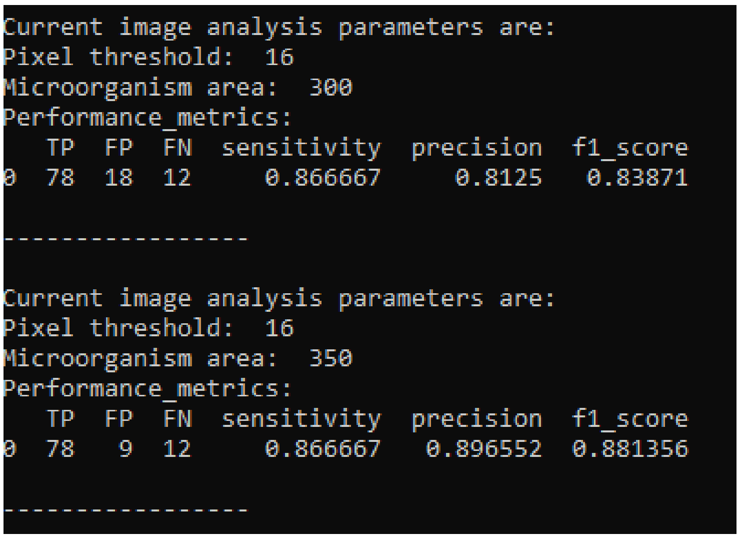
Fig. 19. Example image with the different categories of particle types. Note that this image does not display the full quadrant.

4) **Update your final_particles file file in a text editor**

- Remove all false positive detections (i.e., lines with particle category = 3) and rename/save this final table to "all_females.csv"

5) Running the optimization module

- To run the optimization module, you need
  - The raw movies
  - The all_females.csv files corresponding to each raw movie, which contain the 'ground truth' information (i.e., the coordinates of all particles that can be detected)
  - A copy of your wellcounter_config.yml file (Store the original version at a safe place, as the optimization module will alter the config file)
- Edit the optimization parameters in your wellcounter_config.yml file under the section "optimize":
  - The start value, end value, and step size of each optimization parameter
  - Paths to training data and a directory for the output of optimization (optimize_imaging_parameters_<date>.csv)
  - Save the configuration file
- Run the optimization module with the command `python wellcounter_optimization_module.py`. During analysis, the screen output will display the current parameter settings and their associated performance metrics (**Fig. 20**). All these data will also be saved in the file optimize_imaging_parameters_<date>.csv.



```
Current image analysis parameters are:
Pixel threshold:  16
Microorganism area:  300
Performance_metrics:
   TP  FP  FN  sensitivity  precision  f1_score
0  78  18  12     0.866667     0.8125   0.83871

----------------

Current image analysis parameters are:
Pixel threshold:  16
Microorganism area:  350
Performance_metrics:
   TP  FP  FN  sensitivity  precision  f1_score
0  78   9  12     0.866667   0.896552  0.881356

----------------
```

Fig. 20. Screen output of the optimization module.