Cell-ACDC

Release 1.4.31

Francesco Padovani and Benedikt Mairhoermann

May 07, 2024

CONTENTS

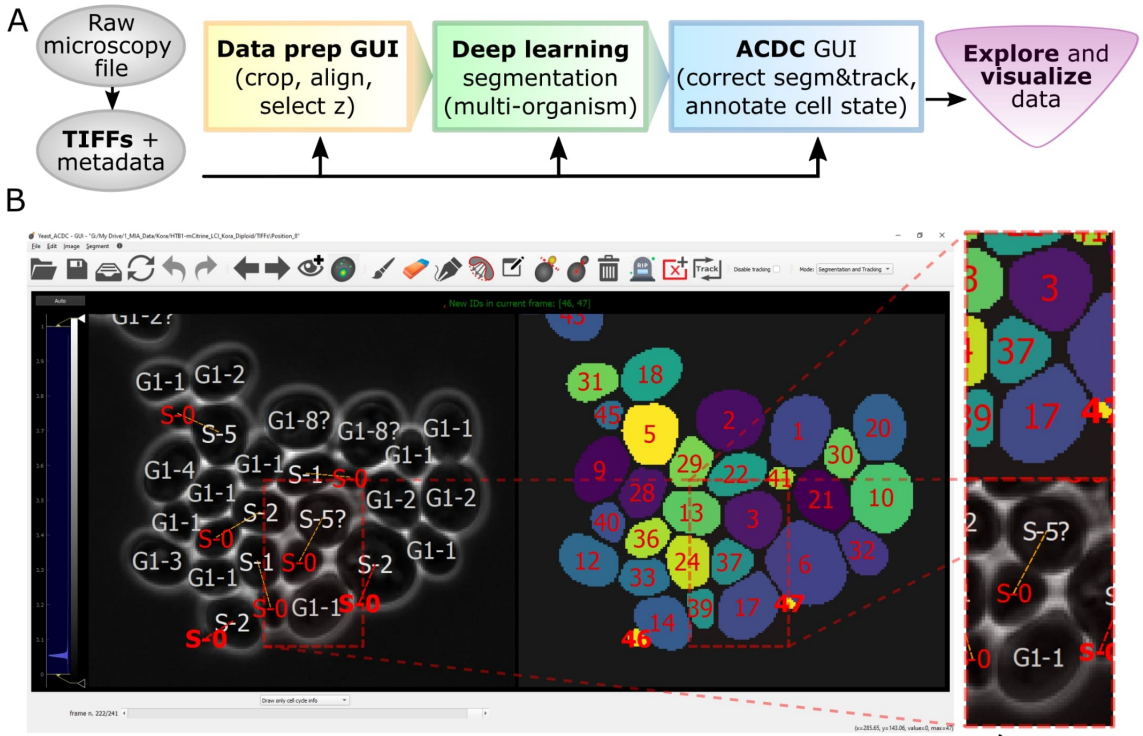
CHAPTER

ONE

A GUI-BASED PYTHON FRAMEWORK FOR SEGMENTATION, TRACKING, CELL CYCLE ANNOTATIONS AND QUANTIFICATION OF MICROSCOPY DATA

 Source code on GitHub

Written in Python 3 by Francesco Padovani *and* Benedikt Mairhoermann.



frame_n	Cell_ID	cell_cycle_stage	generation_num	relative_ID	relationship	emerg_frame_i	division_frame_i	is_history_known
218	3	S/G2/M	5	37	mother	-1	195	FALSE
218	37	S/G2/M	0	3	bud	218	-1	TRUE
218	17	G1	1	6	mother	156	195	TRUE

Overview of pipeline and GUI

CHAPTER

TWO

CONTENTS

2.1 Overview

Let's face it, when dealing with segmentation of microscopy data we often do not have time to check that **everything is correct**, because it is a **tedious** and **very time consuming process**. Cell-ACDC comes to the rescue! We combined the currently **best available neural network models** (such as Segment Anything Model (SAM), YeaZ, cellpose, StarDist, YeastMate, omnipose, delta, DeepSea, etc.) and we complemented them with a **fast and intuitive GUI**.

We developed and implemented several smart functionalities such as **real-time continuous tracking**, **automatic propagation** of error correction, and several tools to facilitate manual correction, from simple yet useful **brush** and **eraser** to more complex flood fill (magic wand) and Random Walker segmentation routines.

See below **how it compares** to other popular tools available (*Table 1 of our* publication).

	Cell-ACDC	YeaZ[4]	Cellpose[5]	YeastMate[6]	DeepCell[24]	PhyloCell[13]	CellProfiler[14]	ImageJ/Fiji[11]	YeastSpotter[22]	YeastNet[23]	MorphoLibJ[12]
Deep-learning segmentation	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗
Traditional segmentation	✓	✗	✗	✗	✗	✓	✓	✓	✗	✗	✓
Tracking	✓	✓	✗	✗	✓	✓	✓	✓	✗	✗	✗
Supports manual corrections	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓
Automatic handling of real-time tracking	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Automatic propagation of correction	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
Automatic mother-bud pairing	✓	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗
Pedigree annotation	✓	✗	✗	✓	✓	✓	✓	✓	✗	✗	✗
Cell division annotation	✓	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗
Downstream analysis	✓	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗
Supports 3D z-stacks	✓	✗	✓	✗	✓	✗	✓	✓	✗	✗	✓
Supports multiple model organisms	✓	✗	✓	✗	✓	✗	✓	✓	✗	✗	✓
Supports Bio-formats	✓	✗	✗	✗	✗	✗	✓	✓	✗	✗	✓
User manual	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
Open source	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Does not require a licence	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓

2.1.1 Is it only about segmentation?

Of course not! Cell-ACDC automatically computes **several single-cell numerical features** such as cell area and cell volume, plus the mean, max, median, sum and quantiles of any additional fluorescent channel's signal. It even performs background correction, to compute the **protein amount and concentration**.

You can load and analyse single **2D images**, **3D data** (3D z-stacks or 2D images over time) and even **4D data** (3D z-stacks over time).

Finally, we provide Jupyter notebooks to **visualize** and interactively **explore** the data produced.

Do not hesitate to contact me here on GitHub (by opening an issue) or directly at my email padovaf@tcd.ie for any problem and/or feedback on how to improve the user experience!

Bidirectional microscopy shift error correction

Is every second line in your files from your bidirectional microscopy shifted? Look here for further information on how to correct your data.

2.2 Citation

If you use Cell-ACDC in your publication, please cite:

> Francesco Padovani, Benedikt Mairhörmann, Pascal Falter-Braun, Jette Lengefeld, and Kurt M. Schmoller *Segmentation, tracking and cell cycle analysis of live-cell imaging data with Cell-ACDC*. BMC Biol 20, 174 (2022) https://doi.org/10.1186/s12915-022-01372-6

Important: Wwhen citing Cell-ACDC make sure to also cite the paper of the segmentation models and trackers you used! Below you find the links to the models currently available in Cell-ACDC.

2.2.1 Segmentation models

- Cellpose v2
- Cellpose v3
- YeaZ
- StarDist
- Segment Anything Model (SAM)
- Delta
- DeppSea
- Omnipose
- YeastMate

2.2.2 Trackers

- Delta
- DeepSea
- TAPIR
- Bayesian tracker (btrack)
- Trackpy

2.3 Installation

Here you will find a detailed guide on how to install Cell-ACDC. In general, you should be fine with installing the stable version, however, Cell-ACDC development is still quite rapid and if you want to try out the latest features we recommend installing the latest version. On the other hand, installing from source is required only if you plan to contribute to Cell-ACDC development. In that case see this section *How to contribute to Cell-ACDC*.

Tip: If you are **new to Python** or you need a **refresher** on how to manage scientific Python environments, I highly recommend reading this guide by Dr. Robert Haase.

2.3.1 Install stable version

1. **Install Anaconda or Miniconda**
   Anaconda is the standard **package manager** for Python in the scientific community. It comes with a GUI for user-friendly package installation and management. However, here we describe its use through the terminal. Miniconda is a lightweight implementation of Anaconda without the GUI.
2. **Open a terminal**
   Roughly speaking, a terminal is a **text-based way to run instructions**. On Windows, use the **Anaconda prompt**, you can find it by searching for it. On macOS or Linux you can use the default Terminal app.
3. **Update conda** by running the following command:

   ```
   conda update conda
   ```

   This will update all packages that are part of conda.
4. **Create a virtual environment** with the following command:

   ```
   conda create -n acdc python=3.10
   ```

   This will create a virtual environment, which is an **isolated folder** where the required libraries will be installed. The virtual environment is called `acdc` in this case.
5. **Activate the virtual environment** with the following command:

   ```
   conda activate acdc
   ```

This will activate the environment and the terminal will know where to install packages. If the activation of the environment was successful, this should be indicated to the left of the active path (you should see `(acdc)` before the path).

Important: Before moving to the next steps make sure that you always activate the `acdc` environment. If you close the terminal and reopen it, always run the command `conda activate acdc` before installing any package. To know whether the right environment is active, the line on the terminal where you type commands should start with the text `(acdc)`, like in this screenshot:

Windows

macOS

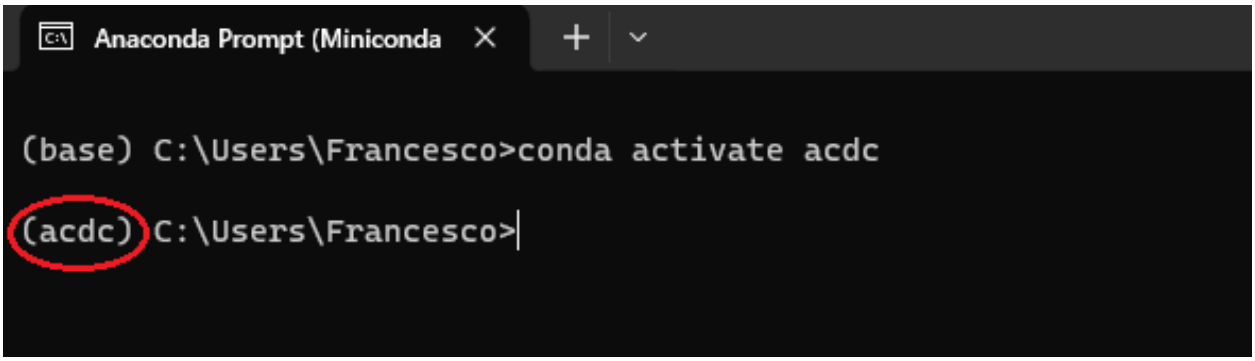


Fig. 1: Anaconda Prompt after activating the `acdc` environment with the command `conda activate acdc`.

Fig. 2: Terminal app after activating the `acdc` environment with the command `conda activate acdc`.

---

6. **Update pip** with the following command:

```
python -m pip install --upgrade pip
```

While we could use conda to install packages, Cell-ACDC is not available on conda yet, hence we will use `pip`. Pip the default package manager for Python. Here we are updating pip itself.

7. **Install Cell-ACDC** with the following command:

```
pip install "cellacdc"
```

This tells pip to install Cell-ACDC.

8. **Install the GUI libraries**:

   After successful installation, you should be able to **run Cell-ACDC with the command** `acdc`. Remember to **always activate** the `acdc` environment with the command `conda activate acdc` every time you open a new terminal before starting Cell-ACDC.

   The first time you run Cell-ACDC you will be guided through the automatic installation of the GUI libraries. Simply answer `y` in the terminal when asked.

   At the end you might have to re-start Cell-ACDC.

   See also:

   If you prefer to **install the GUI libraries manually**, these are the pacakges required:

   conda

   pip

```
conda install -c conda-forge pyqt
conda install -c conda-forge qtpy
conda install -c conda-forge pyqtgraph
conda install -c conda-forge seaborn
conda install -c conda-forge pytables
```

```
pip install PyQt6==6.6.0 PyQt6-Qt6==6.6.0
pip install qtpy
pip install pyqtgraph
pip install seaborn
pip install tables
```

Updating to the latest stable version of Cell-ACDC

To update to the latest version of Cell-ACDC , open the terminal, activate the `acdc` environment with the command `conda activate acdc` and the run the follwing command:

```
pip install --upgrade cellacdc
```

2.3.2 Install latest version

1. **Install Anaconda or Miniconda**
   Anaconda is the standard **package manager** for Python in the scientific community. It comes with a GUI for user-friendly package installation and management. However, here we describe its use through the terminal. Miniconda is a lightweight implementation of Anaconda without the GUI.
2. **Open a terminal**
   Roughly speaking, a terminal is a **text-based way to run instructions**. On Windows, use the **Anaconda prompt**, you can find it by searching for it. On macOS or Linux you can use the default Terminal app.
3. **Update conda** by running the following command:

   ```
   conda update conda
   ```

   This will update all packages that are part of conda.
4. **Create a virtual environment** with the following command:

```
conda create -n acdc python=3.10
```

This will create a virtual environment, which is an **isolated folder** where the required libraries will be installed. The virtual environment is called `acdc` in this case.

5. **Activate the virtual environment** with the following command:

```
conda activate acdc
```

This will activate the environment and the terminal will know where to install packages. If the activation of the environment was successful, this should be indicated to the left of the active path (you should see `(acdc)` before the path).

Important: Before moving to the next steps make sure that you always activate the `acdc` environment. If you close the terminal and reopen it, always run the command `conda activate acdc` before installing any package. To know whether the right environment is active, the line on the terminal where you type commands should start with the text `(acdc)`, like in this screenshot:

Windows

macOS

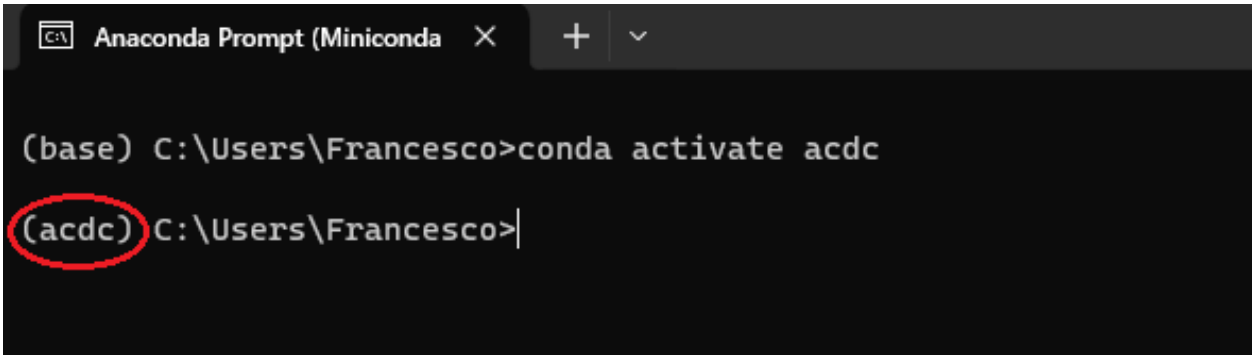


Fig. 3: Anaconda Prompt after activating the `acdc` environment with the command `conda activate acdc`.

Fig. 4: Terminal app after activating the `acdc` environment with the command `conda activate acdc`.

---

6. **Update pip** with the following command:

```
python -m pip install --upgrade pip
```

While we could use conda to install packages, Cell-ACDC is not available on conda yet, hence we will use `pip`. Pip the default package manager for Python. Here we are updating pip itself.

7. **Install Cell-ACDC** directly from the GitHub repo with the following command:

```
pip install "git+https://github.com/SchmollerLab/Cell_ACDC.git"
```

Tip: If you **already have the stable version** and you want to upgrade to the latest version run the following command instead:

```
pip install --upgrade "git+https://github.com/SchmollerLab/Cell_ACDC.git"
```

This tells pip to install Cell-ACDC.

Important: On Windows, if you get the error `ERROR: Cannot find the command 'git'` you need to install `git` first. Close the terminal and install it from here. After installation, you can restart from here, but **remember to activate the** `acdc` **environment first** with the command `conda activate acdc`.

---

8. **Install the GUI libraries**:

After successful installation, you should be able to **run Cell-ACDC with the command** `acdc`. Remember to **always activate** the `acdc` environment with the command `conda activate acdc` every time you open a new terminal before starting Cell-ACDC.

The first time you run Cell-ACDC you will be guided through the automatic installation of the GUI libraries. Simply answer `y` in the terminal when asked.

At the end you might have to re-start Cell-ACDC.

See also:

If you prefer to **install the GUI libraries manually**, these are the pacakges required:

conda

pip

```
conda install -c conda-forge pyqt
conda install -c conda-forge qtpy
conda install -c conda-forge pyqtgraph
conda install -c conda-forge seaborn
conda install -c conda-forge pytables
```

```
pip install PyQt6==6.6.0 PyQt6-Qt6==6.6.0
pip install qtpy
pip install pyqtgraph
pip install seaborn
pip install tables
```

Updating to the latest version of Cell-ACDC

To update to the latest version of Cell-ACDC , open the terminal, activate the `acdc` environment with the command `conda activate acdc` and the run the follwing command:

```
pip install --upgrade "git+https://github.com/SchmollerLab/Cell_ACDC.git"
```

2.3.3 Install from source (developer version)

If you want to try out experimental features (and, if you have time, maybe report a bug or two :D), you can install the developer version from source as follows:

1. **Install Anaconda or Miniconda**
   Anaconda is the standard **package manager** for Python in the scientific community. It comes with a GUI for user-friendly package installation and management. However, here we describe its use through the terminal. Miniconda is a lightweight implementation of Anaconda without the GUI.

2. **Open a terminal**
   Roughly speaking, a terminal is a **text-based way to run instructions**. On Windows, use the **Anaconda prompt**, you can find it by searching for it. On macOS or Linux you can use the default Terminal.

3. **Clone the source code** with the following command:

   ```
   git clone https://github.com/SchmollerLab/Cell_ACDC.git
   ```

   Important: On Windows, if you get the error `ERROR: Cannot find the command 'git'` you need to install `git` first. Close the terminal and install it from here. After installation, you can restart from here, but **remember to activate the `acdc` environment first** with the command `conda activate acdc`.

   ---

4. **Navigate to the Cell_ACDC folder** with the following command:

   ```
   cd Cell_ACDC
   ```

   The command `cd` stands for "change directory" and it allows you to move between directories in the terminal.

5. **Update conda** with the following command:

   ```
   conda update conda
   ```

   This will update all packages that are part of conda.

6. Create a **virtual environment** with the following command:

   ```
   conda create -n acdc python=3.10
   ```

   This will create a virtual environment, which is an **isolated folder** where the required libraries will be installed. The virtual environment is called `acdc` in this case.

7. **Activate the virtual environment** with the following command:

   ```
   conda activate acdc
   ```

This will activate the environment and the terminal will know where to install packages. If the activation of the environment was successful, this should be indicated to the left of the active path (you should see `(acdc)` before the path).

Important: Before moving to the next steps make sure that you always activate the `acdc` environment. If you close the terminal and reopen it, always run the command `conda activate acdc` before installing any package. To know whether the right environment is active, the line on the terminal where you type commands should start with the text `(acdc)`, like in this screenshot:

Windows

macOS

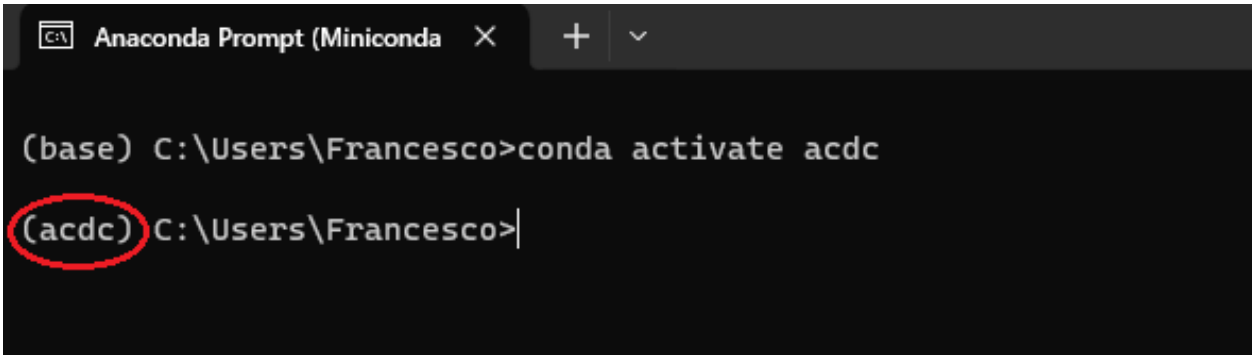


Fig. 5: Anaconda Prompt after activating the `acdc` environment with the command `conda activate acdc`.

Fig. 6: Terminal app after activating the `acdc` environment with the command `conda activate acdc`.

---

8. **Update pip** with the following command:

```
python -m pip install --upgrade pip
```

While we could use conda to install packages, Cell-ACDC is not available on conda yet, hence we will use `pip`. Pip the default package manager for Python. Here we are updating pip itself.

9. **Install Cell-ACDC** with the following command:

```
pip install -e "."
```

The `.` at the end of the command means that you want to install from the current folder in the terminal. This must be the `Cell_ACDC` folder that you cloned before.

10. **Install the GUI libraries**:

    After successful installation, you should be able to **run Cell-ACDC with the command** `acdc`. Remember to **always activate** the `acdc` environment with the command `conda activate acdc` every time you open a new terminal before starting Cell-ACDC.

    The first time you run Cell-ACDC you will be guided through the automatic installation of the GUI libraries. Simply answer `y` in the terminal when asked.

    At the end you might have to re-start Cell-ACDC.

    See also:

    If you prefer to **install the GUI libraries manually**, these are the pacakges required:

    conda

    pip

```
conda install -c conda-forge pyqt
conda install -c conda-forge qtpy
conda install -c conda-forge pyqtgraph
conda install -c conda-forge seaborn
conda install -c conda-forge pytables
```

```
pip install PyQt6==6.6.0 PyQt6-Qt6==6.6.0
pip install qtpy
pip install pyqtgraph
pip install seaborn
pip install tables
```

Updating Cell-ACDC installed from source

To update Cell-ACDC installed from source, open a terminal window, navigate to the Cell-ACDC folder with the command `cd Cell_ACDC` and run `git pull`.

Since you installed with the `-e` flag, pulling with `git` is enough.

2.4 Folder structure

To load data into Cell-ACDC you need to structure the files in a specific way.

In the section *Creating Data Structure* we explain how to achieve this in an automatic fashion.

Here, instead we will focus on how the folder structure should look like.

Important: We do not recommend creating the folder structure manually. Here we only want to provide more details about how the folder structure look like. To create the folder structure automatically from a microscopy file use either module 0 of Cell-ACDC (explained here *Creating Data Structure*) or Fiji macros that you can find *here <https://github.com/SchmollerLab/Cell_ACDC/tree/main/FijiMacros>_*.

In Cell-ACDC we refer to the folders with two names:

1. The experiment folder with an arbitrary name
2. The position folder with names like `Position_1`, `Position_2`, etc.

The experiment folder is a folder that typically identifies a specific experiment and it contains one or more position folders.

The position folder is a folder inside the experiment folder that must contain a folder called `Images`. In Cell-ACDC we refer to "positions" with the same meaning of the term "series" in ImageJ/Fiji. These are also called "Tiles" by some microscopy manufacturers. You can have as many position folders as you like in each experiment folder.

The `Images` folder contains the image files. You will need to create one TIFF file per channel. Each TIFF file can be 2D, 3D (z-stack or 2D over time), or 4D (z-stack over time). You can have as many channels as you want.

The filenames of each file must all start with the same <`basename`>, which is a text that is common to all the files in the folder.

This is probably more clear with an example. Let's say that you have an experiment called `mitochondria_medium_switch` with 5 positions and each position has three channels: `phase_contrast`, `GFP`, and `mCitrine`. With this structure you create a folder called `mitochondria_medium_switch` and inside this folder you create 5 folders, one for each position, called `Position_1`, `Position_2`, ..., and `Position_5`. Inside each position folder you create a folder called `Images`. Inside each `Images` folder you create the TIFF files, one for each channel, all starting with the same basename. The basename is a text that allows you to identify what the experiment is about, for this example we will use the basename `ASY015_mitochondria_medium_switch_`.

In this hypothetical example, the folder structure wood look like the following:

```
mitochondria_medium_switch
├── Position_26
│   └── Images
│       ├── ASY015_mitochondria_medium_switch_metadata.csv
│       ├── ASY015_mitochondria_medium_switch_GFP.tif
│       ├── ASY015_mitochondria_medium_switch_mCitrine.tif
│       └── ASY015_mitochondria_medium_switch_phase_contrast.tif
│
├── Position_2
│   └── Images
│       ...
│
├── ...
│
└── Position_5
    └── Images
        ...
```

You probably have noticed an additional file that ends with `_metadata.csv`. If this file is missing it will be created by Cell-ACDC. It contains useful metadata like the pixel size and other metadata, but it also contains an entry called `basename`. If you don't create this file Cell-ACDC will try to guess the `basename` from the filenames. While this usually works fine, it is better to create this `_metadata.csv` file with the following content:

```
Description,values
basename,ASY015_mitochondria_medium_switch_
```

This way Cell-ACDC will not try to guess the basename and you will avoid weird naming of additional files due to the wrong basename being guessed.

2.5 From 0 to Cell-ACDC mastery: A complete guide

This guide should provide you with everything that you need to know about Cell-ACDC to become a true master. In the future, a abridged version of this guide as well as video tutorials will be added. During several points during the usage of Cell-ACDC, the program may become unresponsive. **DO NOT CLOSE CELL-ACDC**. Instead, check the console for progress or error massages.

Contents

2.5.1 Running Cell-ACDC

1. Open a **terminal** (on Windows use the **Anaconda Prompt** if you installed with `conda`)
2. **Activate** the **environment** (conda: `conda activate acdc`, pip on Windows: `.\env\Scripts\activate`, pip on Unix: `source env/bin/activate`)
3. **Run** the command `acdc` or `cellacdc`

2.5.2 The Main Menu

The main menu is a **hub** through which you can access all relevant modules.

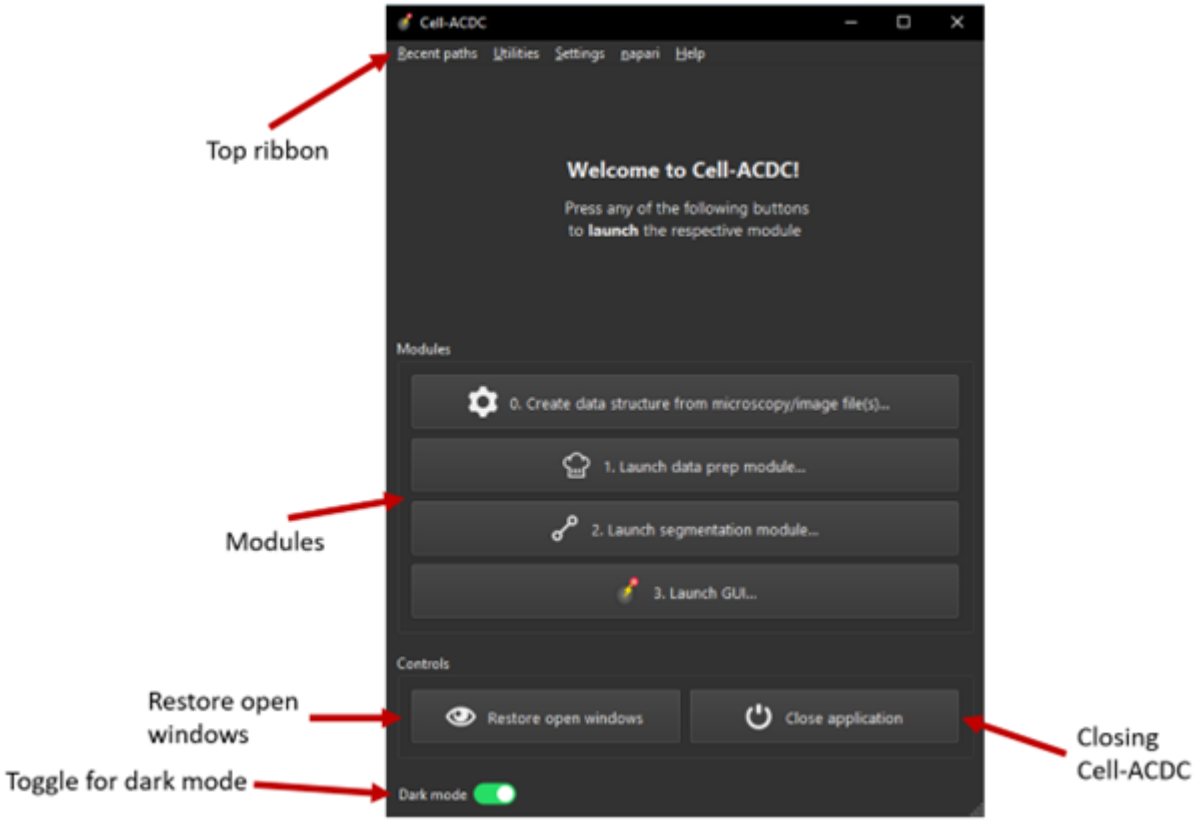


Fig. 7: Overview of the main menu

Module buttons

Through the main menu, all modules of Cell-ACDC can be accessed:

1. **Create data structure from microscopy/image file(s)…**
   This module will allow you to **create a suitable data structure** from raw microscopy files.
2. **Launch data prep module…**
   With this module you can **align** time-lapse microscopy data and select a **sharp image from a z-stack**, as well as **crop** images.
3. **Launch segmentation module…**
   Using this module, you can perform **segmentation** and **tracking** tasks using common methods.
4. **Launch GUI…**
   Lastly, using the GUI the resulting data can be **inspected** and **corrected** for mistakes. Also, **cell cycle annotation** can be carried out.

Top ribbon options include:

- **Recent paths**
  - Here you can access a **history of recent paths** used in any of the modules.
- **Utilities**
  - Some **extra utilities** which might be useful, including:
    - Convert file formats
    - Segmentation: Creating 3D segmentation masks
    - Tracking: Tracking and counting sub-cellular objects as well as applying tracking information from tabular data
    - Measurements: Tools for computing measurements
    - Add lineage tree table to one or more experiments
    - Concatenate acdc output tables from multiple positions
    - Create required data structure from image files
    - Re-apply data prep steps to selected channels
    - Align or revert alignment
    - Rename files by appending additional text
- **Settings**
  - Allows manipulation of the **user profile** path.
- **Napari**
  - View the Napari lineage tree.
- **Help**
  - Provides links to **user manuals and start up guide**, as well as a link to the **relevant paper** for citation and guides on how to contribute, viewing the **log files** and show **additional info** about Cell-ACDC, including the **installation path**.

Additional options include:

- **Restoring** all open windows

  Restores all windows which were minimized
- **Closing** the application and all active modules
- Toggle switch for **dark mode**

2.5.3 Creating Data Structure

0. Create data structure from microscopy/image file(s)…

The first step in analysing data with Cell-ACDC is creating a **suitable data structure** from raw microscopy files. This can be done c ompletely automated using module 0.

> **Warning:** If you are on **MacOS** you will need to use the **ImageJ/Fiji macros** to create the data structure. See the section *Create Data Structure with ImageJ/Fiji macros* for details about how to use the macros.
>
> This is because the library `python-bioformats` does not work on MacOS. We are working on alternatives, but in the meantime the Fiji macros will work just fine. Thank you for you patience.

To start off, launch the module by pressing on the corresponding button in the main menu.

This will open a window in which you can choose how you want to proceed.

Note: Cell-ACDC can use Bio-Formats or the AICSImageIO libraries to read microscopy files.

Bio-Formats requires Java and a python package called Javabridge, that will be automatically installed if missing. We recommend using Bio-Formats, since it can read the metadata of the file, such as pixel size, numerical aperture etc.

If Bio-Formats fails, try using AICSImageIO.

Alternatively, if you already pre-processed your microscopy files into TIF files, you could choose to simply re-structure them into the Cell-ACDC compatible format.

After choosing an option, another window will open prompting you to select **what kind of data** you want to extract from the raw microscopy file:

- Single microscopy file with multiple positions
- One or more microscopy files, one for each position
- One or more microscopy files, one for each channel
- NONE of the above

Please select the appropriate option. Afterwards, you are prompted to **create an empty folder** in which only the microscopy file(s) are present. After doing so, select "Done". Next, you will be prompted to select this folder. After selecting the **destination folder**, which by default is the folder you selected in the step before, Cell-ACDC will attempt to load OEM metadata.

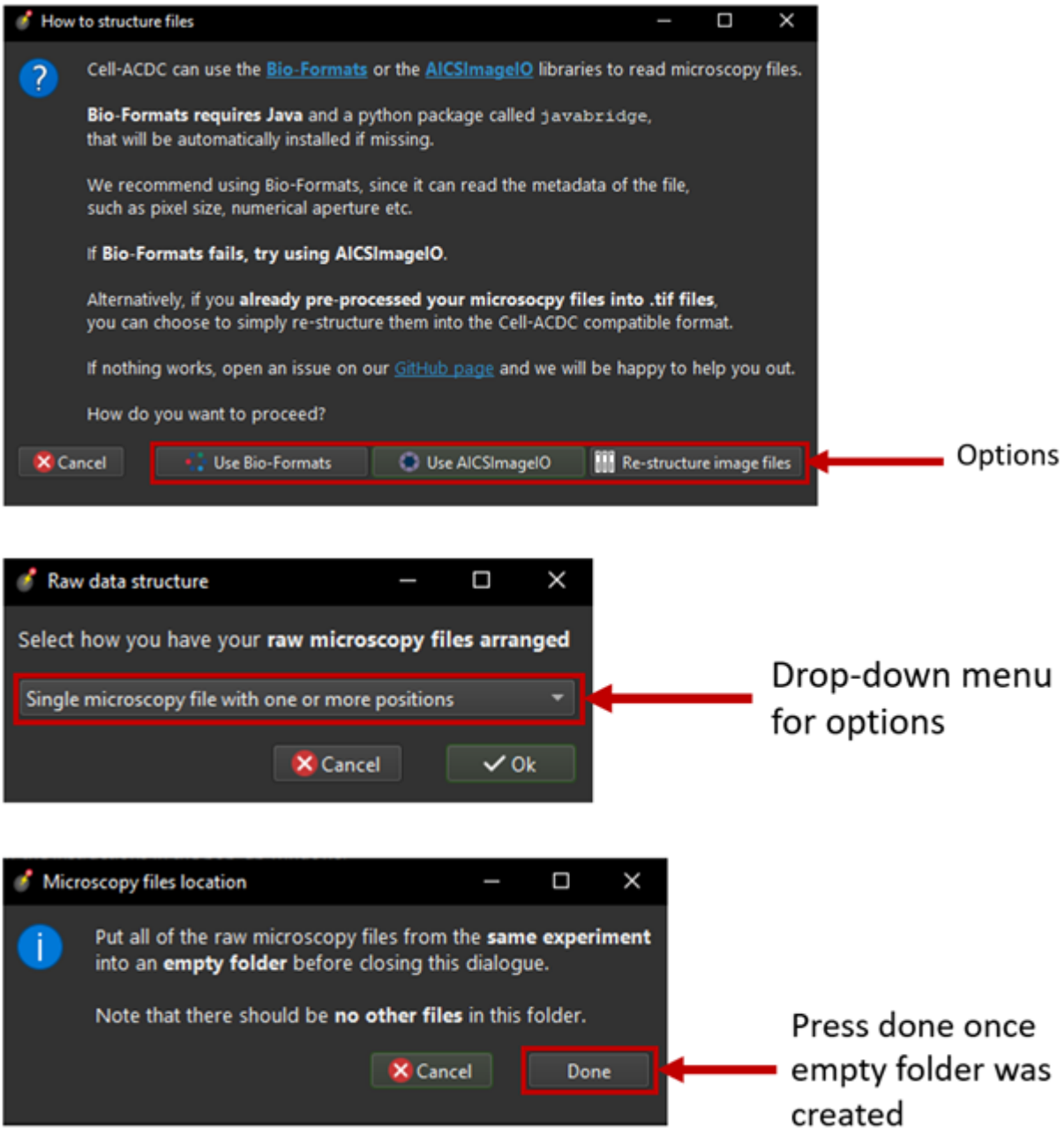
A window with the extracted metadata should appear, which may **take a few seconds to load**. Make sure to **double check** all values and **change "Order of Dimensions"** to the appropriate value. To double **check if the dimensions are in the correct order**, select the eye icon () next to "Channel 0" and use the scrollbars to go through the z-coordinate and time-coordinate. Once all values are in order, press "Ok". If the values are the same for all positions, feel free to click "Use the above metadata for all the next positions".

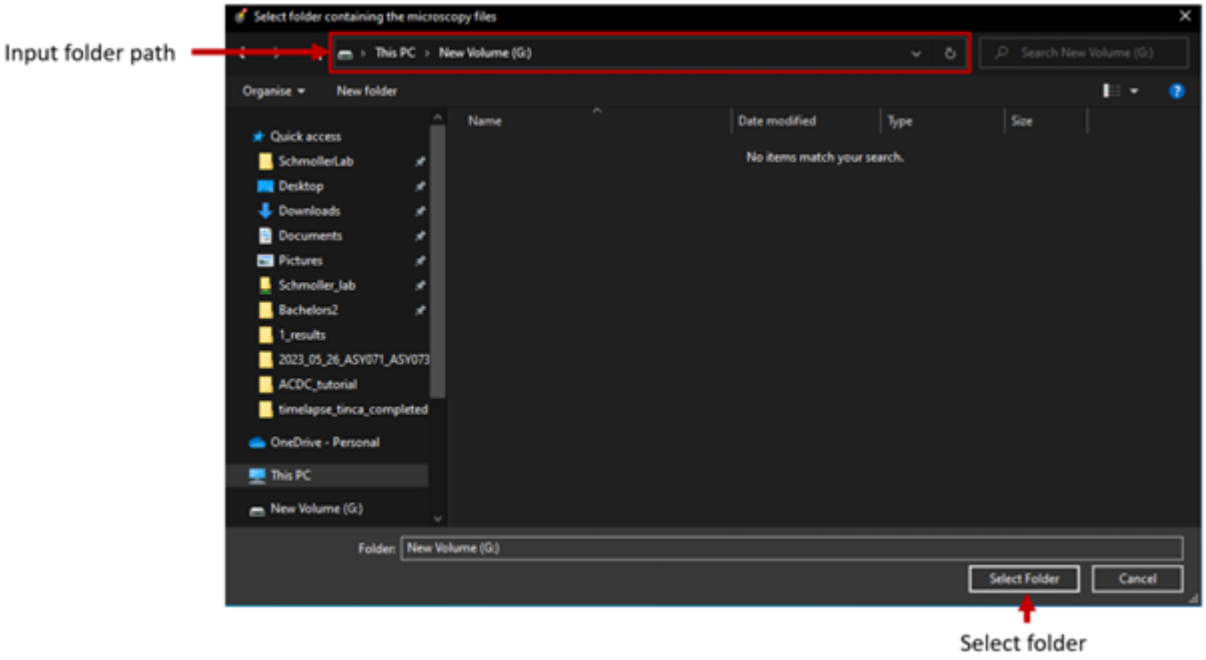
Note: that if you have several files, and you press "Ok" and not one of the two other options, the process will stop after each file, and you need to confirm the metadata again.

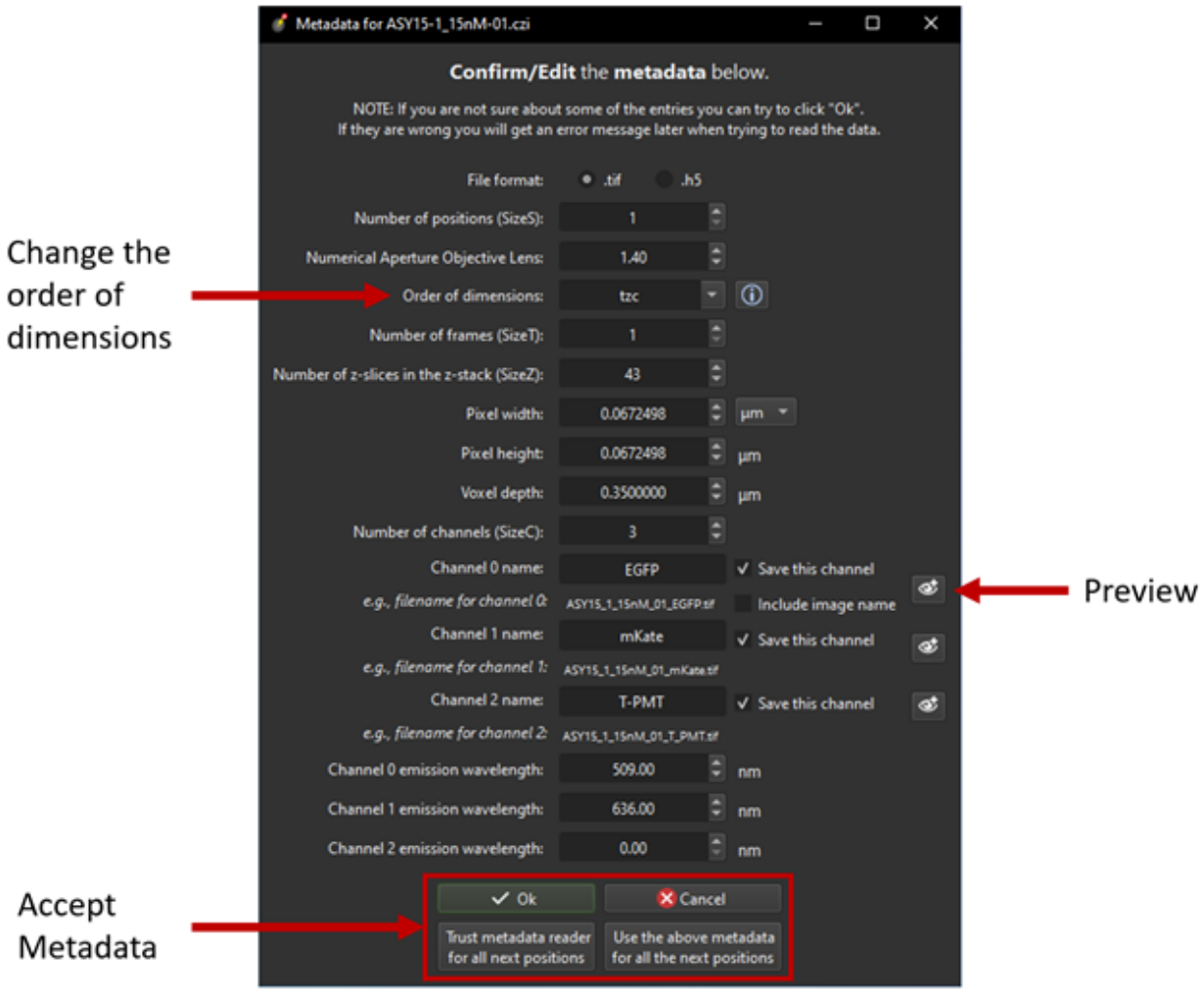
Each position is saved in a separate folder. The metadata are stored both in a TXT and SCV file, while the channels are stored in separate TIF files.

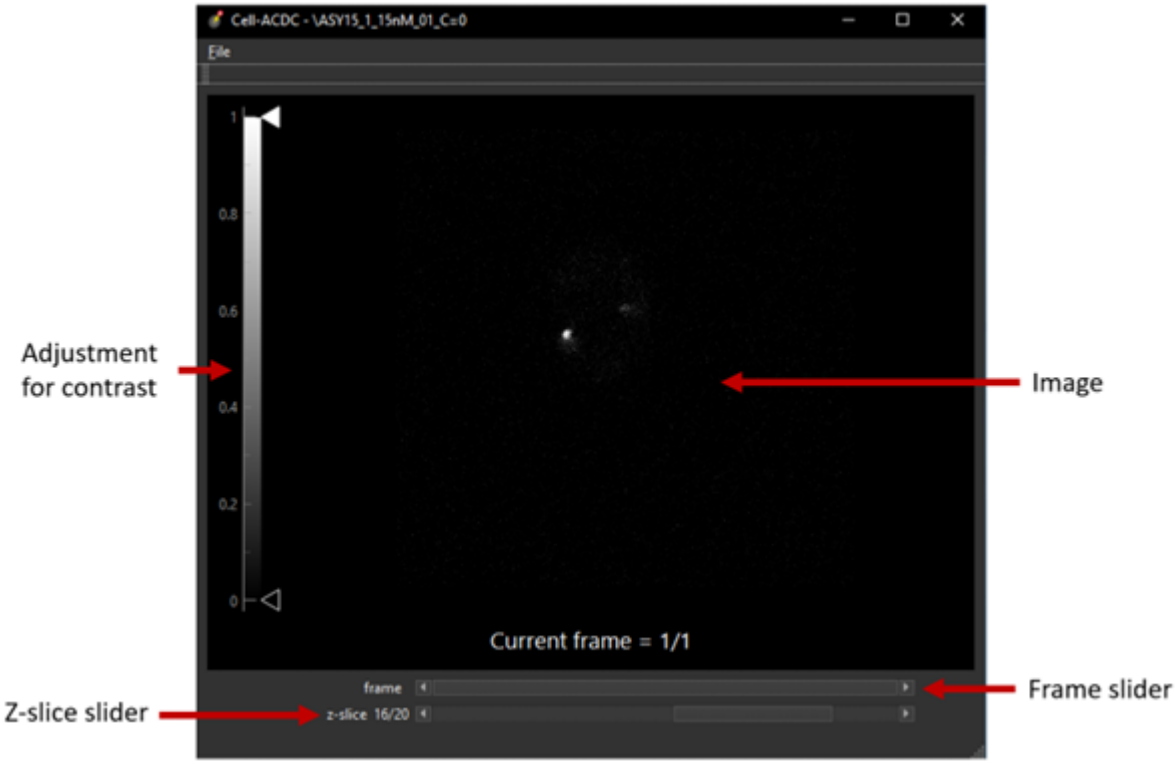
Note: A computer with sufficient RAM is needed in this step! The required amount is heavily reliant on the size of the project.

It is good practice to keep the original files for future reference, even though they are not needed in future steps.




Input folder path
Select folder containing the microscopy files
This PC › New Volume (G:)
Organise
New folder
Quick access
SchmollerLab
Desktop
Downloads
Documents
Pictures
Schmoller_lab
Bachelors2
1_results
2023_05_26_ASY071_ASY073
ACDC_tutorial
timelapse_tinca_completed
OneDrive - Personal
This PC
New Volume (G:)
Name
Date modified
Type
Size
No items match your search.
Folder:
New Volume (G:)
Select Folder
Cancel
Select folder


Change the order of dimensions
Metadata for ASY15-1_15nM-01.czi
Confirm/Edit the metadata below.
NOTE: If you are not sure about some of the entries you can try to click "Ok".
If they are wrong you will get an error message later when trying to read the data.
File format:
.tif
.h5
Number of positions (SizeS):
1
Numerical Aperture Objective Lens:
1.40
Order of dimensions:
tzc
Number of frames (SizeT):
1
Number of z-slices in the z-stack (SizeZ):
43
Pixel width:
0.0672498
µm
Pixel height:
0.0672498
µm
Voxel depth:
0.3500000
µm
Number of channels (SizeC):
3
Channel 0 name:
EGFP
Save this channel
e.g., filename for channel 0:
ASY15_1_15nM_01_EGFP.tif
Include image name
Channel 1 name:
mKate
Save this channel
e.g., filename for channel 1:
ASY15_1_15nM_01_mKate.tif
Channel 2 name:
T-PMT
Save this channel
e.g., filename for channel 2:
ASY15_1_15nM_01_T_PMT.tif
Channel 0 emission wavelength:
509.00
nm
Channel 1 emission wavelength:
636.00
nm
Channel 2 emission wavelength:
0.00
nm
Preview
Accept Metadata
Ok
Cancel
Trust metadata reader for all next positions
Use the above metadata for all the next positions



Data-structure created by Cell-ACDC

```
Experiment folder
├── Position_1
│   └── Images
│       ├── basename_s01_metadataXML.txt
│       ├── basename_s01_metadata.csv
│       ├── basename_s01_ChannelName1.tif
│       ├── basename_s01_ChannelName2.tif
│       ...
├── Position_2
│   └── Images
│       ├── basename_s01_metadataXML.txt
│       ...
└── Position_n
    └── Images
        ├── basename_s0n_metadataXML.txt
        ...
```

2.5.4 Preparing data for further analysis (Data-prep)

1. Launch data prep module…

Through pressing "Launch data prep module…" in the main menu, the module can be launched. In this step you can:

a) Select a z-slice or z-projection for segmentation of 3D z-stacks.

b) Align frames of time-lapse microscopy data (RECOMMENDED, it is revertible).

c) Calculate background metrics (median, mean etc.) from one or more rectangular areas. The median will be used later for background subtraction. The areas are movable and resizable.

d) Select a region of interest (ROI) for segmentation.

e) Crop images to reduce memory usage (RECOMMENDED, if possible).

The alignment process is done using the function `skimage.registration.phase_cross_correlation` from the scikit-image library.

To start off, click **"File"** in the top ribbon and then select **"Open"**. Select the position folder, for example "Position_1", which you want to start preparing. A pop up will appear which asks you for the channel name. Here you should input the channel on which **basis you want to align**.

In the next menu, select the **desired number of frames and z-slices**. Here you can also add another custom field, which will be saved in the metadata table. Later, this will be added as a column to the output table.

Next, go through each frame and **select the z-slice which is the sharpest** (if your data is 3D). Using the **buttons in the top button row**, you can apply the current slice to all future () or past () frames, as well as apply a gradient () from the current frame to the first one. The selection is saved automatically in (almost) real time. If you only need to do this step, feel free to close the window after finishing.

Alternatively, a projection can be used. This is done through the projection drop down menu in the bottom right.

Next, select **"start"** () from the buttons bar. This will **start the alignment process**. **The window may become unresponsive**, please check the terminal for progress.

Note: Do this even if you don't have a time lapse experiment, as it allows you to carry on to the next step and won't change the data.

For time-lapse microscopy you can load only one position at a time. Select multiple positions only if you have single 3D z-stacks or single 2D images.

Afterwards, the **region of interest (ROI)** as well as the **background ROI (Bkgr. ROI)** can be adjusted. This is done through drag and drop on the edges and resizing on the turquoise rhombuses. Make sure that the ROI covers all cells of interest on all frames and that the Bkgr. ROI is on an area without cells. **Multiple ROIs** () **and Bkgr. ROIs** () can be added through the corresponding buttons. **Right click** on one of the frames to show an interaction menu through which you can **remove** it.

Tip: The background ROIs are used to compute background values from those areas. These values will be saved in the `acdc_output` CSV table that you can create with the module 3 `Launch GUI...`. The background values specific to these ROIs will be called `<ChannelName>_dataPrepBkgr_bkgrVal_<metric_name>`, where `<metric_name>` will be the `median`, `mean`, etc.

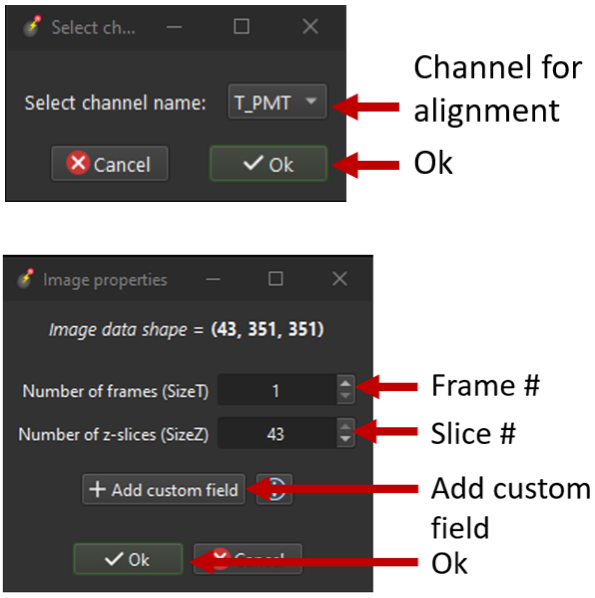
Additionally, the median of the background will be used to compute background corrected measurements like `amount_dataPrepBkgr` (i.e., background corrected sum intensity) and `concentration_dataPrepBkgr`.

If you are working with 3D z-stacks and you want to crop z-slices click on the **"Crop z-slices"** () button. Once all is set, press the **"Crop XY"** () button to preview the crop.

To save the cropped data, click on the **"Save cropped data"** () button. **This will overwrite the previous files**. The window may become **unresponsive**.

Note: If the Bkgr. ROI is not visible, background from data-prep will not be computed. If you want to set a Bkgr. ROI, press the Bkgr. ROI button () to add as many background ROIs as needed.

Data such as the selected frame is stored in segmInfo.csv, while aligned.npz stores the alignment data.



2.5.5 Segmentation and tracking

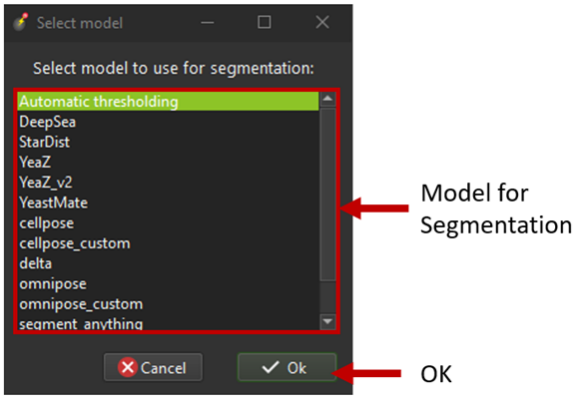
2. Launch segmentation module…

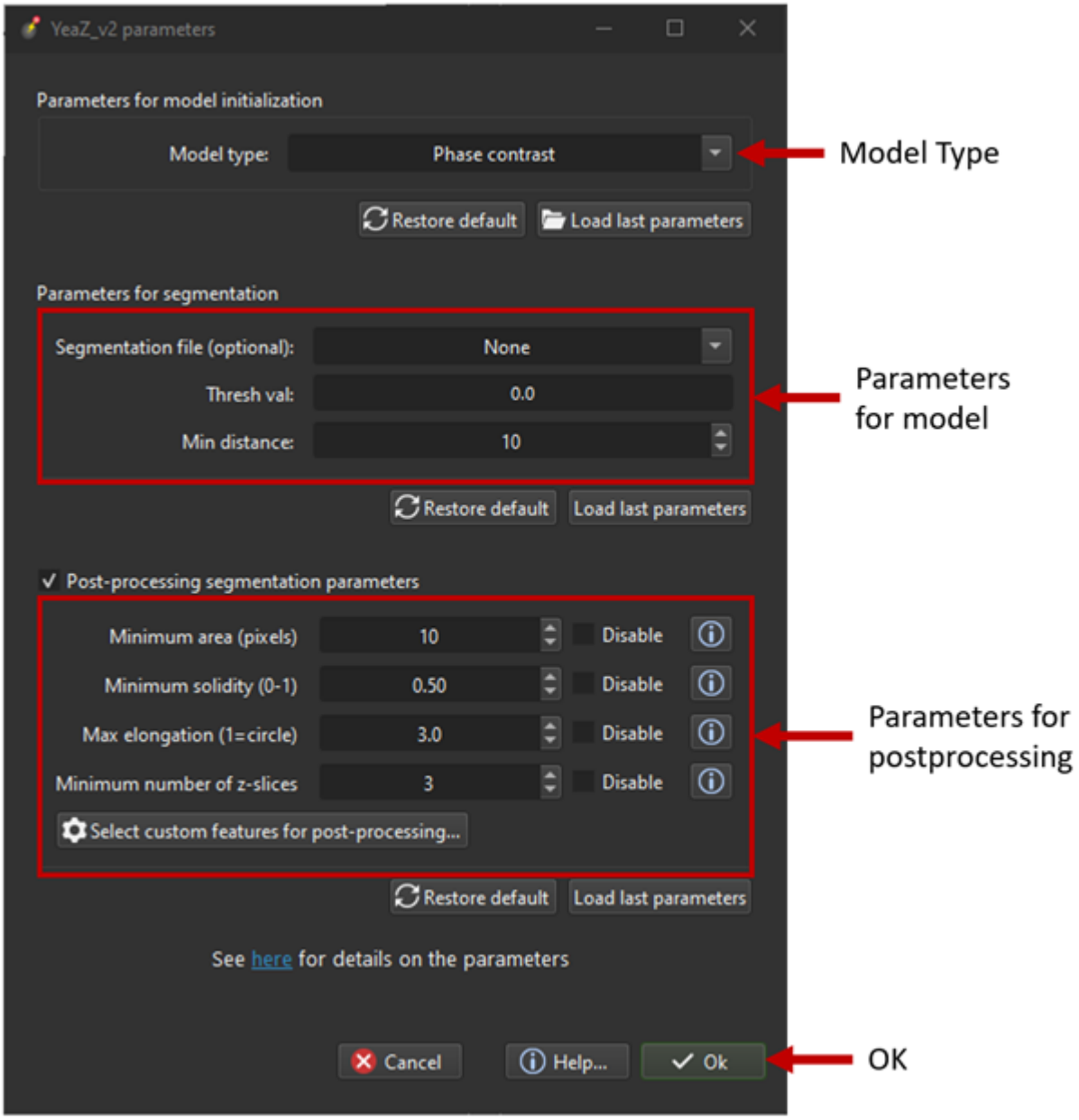
This module can be used to **segment and track objects** in your data. A plethora of options are available already, and new ones are added constantly. You can also add **your own models**, for this please see this guide.

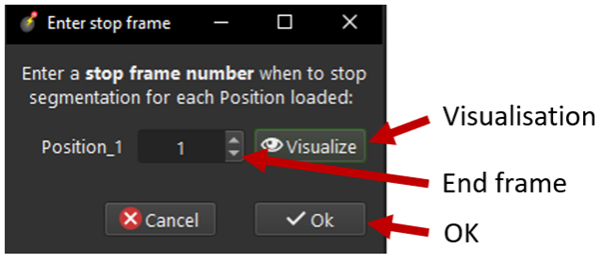
Upon launching the module, you first will be prompted to **select a folder**. This process is the same as before. Next, like before, you are prompted to select a channel which should be **used for segmentation**.

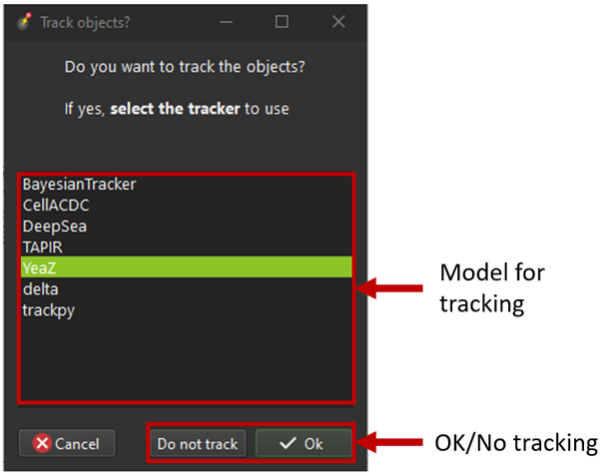
After a short wait, you are prompted to **select the model** you want to use for **segmentation**, after which one needs to confirm the parameters for segmentation as well as post processing.

Next, you can **select a stop frame** if you don't want to segment and track the entire experiment. Lastly, you need to **select the model** which should be used for **tracking**. The process now begins, and you can lay back and watch the computer work for you.


Select model
Select model to use for segmentation:
Automatic thresholding
DeepSea
StarDist
YeaZ
YeaZ_v2
YeastMate
cellpose
cellpose_custom
delta
omnipose
omnipose_custom
segment_anything
Cancel
Ok
Model for Segmentation
OK


YeaZ_v2 parameters
Parameters for model initialization
Model type:
Phase contrast
Restore default
Load last parameters
Parameters for segmentation
Segmentation file (optional):
None
Thresh val:
0.0
Min distance:
10
Restore default
Load last parameters
Post-processing segmentation parameters
Minimum area (pixels)
10
Disable
Minimum solidity (0-1)
0.50
Disable
Max elongation (1=circle)
3.0
Disable
Minimum number of z-slices
3
Disable
Select custom features for post-processing...
Restore default
Load last parameters
See here for details on the parameters
Cancel
Help...
Ok
Model Type
Parameters for model
Parameters for postprocessing
OK





Root folder

- Position_1
  - Images
    - ExperimentName _s01_metadataXML.txt
    - ExperimentName _s01_metadata.csv
    - ExperimentName _s01_dataPrep_bkgrROIs.json
    - ExperimentName _s01_dataPrepROIs_coords.csv
    - ExperimentName _s01_segmInfo.csv
    - ExperimentName _s01_align_shift.npy
    - ExperimentName _s01_segm.npz
    - ExperimentName _s01_custom_annot_params.json
    - ExperimentName _s01_acdc_output.csv
    - ExperimentName _s01_segm_hyperparams.ini
    - ExperimentName _s01_last_tracked_i.txt
    - ExperimentName _s01_segm_hyperparams.ini
    - ExperimentName _s01_custom_combine_metrics.ini
    - ExperimentName _s01_ChannelName1.tif
    - ExperimentName _s01_ChannelName1_aligned.npz
    - ExperimentName _s01_ChannelName1_aligned_bkgrRoiData.npz
- ... ...

2.5.6 Correcting Tracking and Segmentation Mistakes, Cell Cycle Annotation

3. Launching GUI…

The GUI is very useful to review and annotate data. For a full breakdown of all tools, please see the section GUI tools.

Its main functions are:

a) **Test** which **segmentation method** works best for your dataset.

b) **Correct** segmentation and tracking errors.

c) Cell cycle **annotations**.

Usage with time lapse data

For time-lapse data, you can load one position (one video) at a time. With this data, the GUI has three modes that can be toggled from the selector on the toolbar:

1. Viewer mode (default mode, used only for visualisation).
2. Cell cycle analysis mode.
3. Segmentation and tracking mode.

The main idea is that when you visit a frame for the first time, some automatic functions are triggered: tracking in "Segmentation and tracking" mode, mother-bud pairing in "Cell cycle analysis" mode. See the GUI tools section for a run down of all tools.

These functions are not triggered when you visualize a frame that you already visited before.

Usage with snapshot data (no time-lapse)

For snapshot data, you can load multiple positions at the same time. When prompted, simply click on multiple selection button, and then select the positions with `Ctrl+click` for selecting specific positions, or `Shift+click` to select a range, or `Ctrl+A` to select all.

Once loaded, you can navigate through positions with left and right arrow or with the position scrollbar below the left image.

See sections "Segmentation and tracking" and "Cell cycle analysis" for further information. See the GUI tools section for a run down of all tools.

Correcting Tracking and Segmentation Mistakes

The first step in using the GUI is to load a file. For this, click on **"File"** in the top ribbon and select **"Load folder"**, or directly select the corresponding button (). This will open a window which prompts you to select a folder. After selecting the folder containing the information for the position you want to analyse, you will be prompted to **select the channel you want to view** as well as double **check the metadata**.

Alternatively, if only a single image or video should be analysed, select `File` → `Open image/video file...`.

Note: If you load a single image or video file without the required data structure, the Cell-ACDC output will be saved in a sub-folder called `<timestamp>_acdc_output`.

After first loading data, you will notice that the current mode is set to "Viewer". This allows you to freely browse through all images, which can be useful for gaining an overview of the data.

To start editing, change the mode to **"Segmentation and Tracking"**.

Important tools:

- **"Eraser"** and **"Brush"** function as you expect.
- **"Separation**" can be used to s**eparate two cells** which were not segmented properly.
- **"Edit ID"** can be used to **change the ID** of a cell and mend tracking errors.
- **"Merge IDs"** for **merging two IDs** if a cell was segmented into two parts.
- **"Annotate as dead"**, **"exclude from analysis"**, **"deletion region"** and "**delete all objects touching ROI border"** for **excluding cells** or regions from analysis.
- **"Repeat tracking"** and **"repeat segmentation"** for **repeating** the respective processes, which can be used to bring frame in line with previous frames.

Important tips:

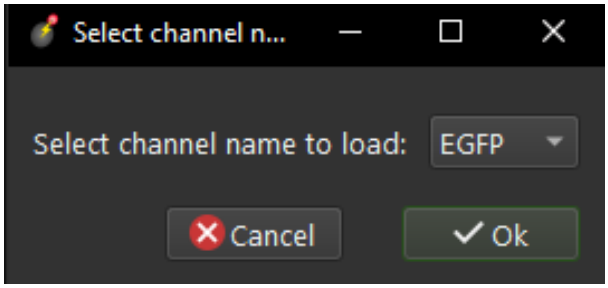
- Cells with a **thick red contour** and **thick ID** are **new cells** which were not present in the previous frame.
- **Yellow contours** with a **yellow ID** and a question mark show the contours of cells which were present in the previous frame but are **missing** in the currently viewed frame.
- Most **key bindings** can be **viewed** and customized via the menu found in the **top ribbon "Settings" menu**.
- **"H"**: **centre** the picture. Double pressing **"H"**: **resets zoom**.
- **"Alt+Click+Drag"**: **pan/move** image
- **middle mouse button** (Windows) or **Cmd+Click** (MacOS): **delete** a cell ID.
- **"Ctrl+P"**: Visualize **cell cycle annotations** in a **table**.
- **"Ctrl+L"**: **Relabel** object IDs sequentially (1,2,3…etc).
- **"Ctrl+F"**: **Search** and **highlight** specific object ID.
- **Right click** on any point in the picture to reveal **more options**. Most importantly, the option to show a **duplicate picture**. This is useful to both view the contours and the segmentation mask.
- **"Spacebar"**: **Hide/show contours** or **segmentation masks** on left image
- **Double tap a binding** for a tool to select the **"empowered" version**, which can **draw over any cells**. Otherwise, tools only influence the cell on which you start drawing.
- **Shift while drawing with the brush** will force a **new ID** creation.
- You can use the **arrow keys** to **navigate** between frames.
- To **test** the available **segmentation models**, use the `Segment` menu.
- To **visualize** the **frames** of time-lapse data in a second window click on the **"Slideshow"** button on the toolbar
- **Personalize settings** such as font Size, overlay colour and text's colour from the `Edit` menu.

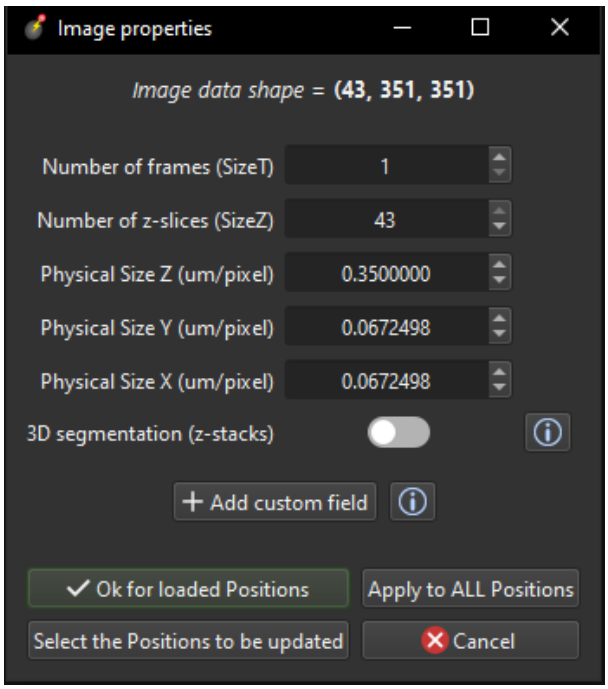
Cell Cycle Annotation

After correcting all errors, change the mode to "Cell Cycle Analysis". You will be presented with a warning that suggests starting from the first frame, which you usually should heed. Important tools for CC-Ana:

- **"Assign bud to mother"** is used if automatic assignment is wrong. For this activate the tool, then press and hold the right mouse button on the bud, then drag to the mother and release.
- **"Annotate unknown history"** can be used to annotate cells which have unknown history.
- **"Reinitialize cell cycle annotation"** for running cell cycle annotation from this frame foreword to make them in line with current edits.
- **"Right click on mother/bud pair"** will **break the bond**. Right click **again** to **rebind** them. This needs to be done manually whenever a mother and bud separate.

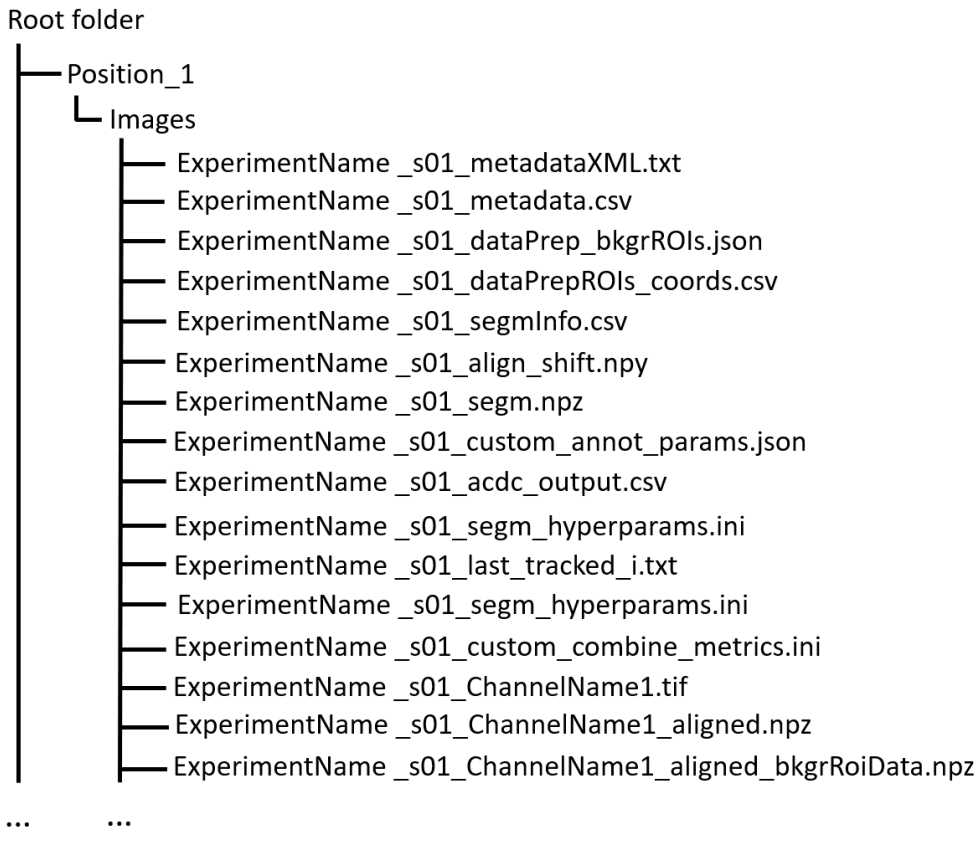
After finishing annotating the first frame, you will be prompted to accept the current annotation. This is only to make sure that the initial annotations are correct.






Top ribbon
File controls
View controls
Edit tools
Mode selection
Warnings
Auto-contrast
Example of missing cell
Object properties menu
Contrast for image
Contrast for segm. mask.
Contrast for image
Example of new Cell
Show Annotation
Show Form
No anno.
Controls for right image
Quick settings
Frame slider
Z-slice slider
Controls for left image
Projection selection


Top ribbon
File controls
View controls
Edit tools
Mode selection
Warnings
Auto-contrast
Unassigned bud
Cell cycle stage
Object properties menu
Contrast for image
Contrast for segm. mask.
Bud
Mother bud bond
Contrast for image
Show Annotation
Show Form
No anno.
Controls for right image
Quick settings
Frame slider
Z-slice slider
Controls for left image
Projection selection



All functions

See the GUI tools section for a full run down of all tools.

`File → Load fluorescent images...`

Used to **load additional images** (e.g., fluorescence signal).

Loaded images will be used to **calculate metrics** such as mean, median, total amount etc. See this section for more details.

`Edit → Smart handling of enabling/disabling tracking`

The GUI has built-in automatic tracking for time-lapse data with the following behaviour:

- If tracking is active (Disable tracking checkbox on the toolbar is UNCHECKED): When you visit a frame that you have never visited before, objects will be **automatically tracked** compared to previous frame.
- If tracking is deactivated (Disable tracking checkbox on the toolbar is CHECKED): When you visit a frame already visited before, it will **not** be tracked.

You can disable this automatic behaviour by unchecking Smart handling of enabling/disabling tracking.

When you disable smart handling, you can enforce tracking on all visited frames no matter if they were previously visited or not. To force this, use the "Disable tracking" checkbox on the toolbar.

Tip: This is useful when you know you have to repeat tracking on already visited frames.

`Image → Normalise intensities → ...`

You can choose to **normalise the intensities** of the displayed images (saved data will not be modified) with the following methods:

- Do not normalise: Displays raw image.
- Convert to floating point format with values [0, 1]: Simply converts image to **floating point**, no normalisation is involved.
- Rescale to [0,1]: Intensities are first converted to **floating point** if needed and then **STRETCHED** to convert the entire [0,1] range.
- Normalize by max value: Intensities are **divided by the max value**.

Shared:

- **Top ribbon:**
  - **File: File manipulation menu with options to load different positions, saving etc.**
    - New
    - Load folder…
    - Open image/video file…
    - Open Recent
    - Load older versions…
    - Save
    - Save as…
    - Save only segme file
    - Load fluorescence images…
    - Load different Position…
    - Exit
  - **Edit: Some edit settings**
    - Customize keyboard shortcuts
    - Text annotation colour
    - Overlay colour
    - Edit cell cycle annotations
    - Smart handling of enabling/disabling tracking
    - Automatic zoom to all cells when pressing "Next/Previous"
  - **View: Some view settings**
    - View cell cycle annotations
    - Show segmentation image
    - Show duplicated left image
  - **Image: Image viewing settings and options**
    - Properties (from config files)
    - Filters

        * Normalize intensities
        * Invert black/white
        * Save labels colormap
        * Randomly shuffle colormap
        * Optimise colormap
        * Zoom to objects (shortcut: H key)
        * Zoom out (shortcut: double press H key)
    - **Segment: Settings for re-segmentation**
        * Segment displayed frame
        * Segment multiple frames
        * Random walker
        * Segmentation post- processing
        * Enable automatic segmentation
        * Relabel IDs sequentially
    - **Tracking: Settings for re-tracking**
        * Select real-time tracking algorithm
        * Repeat tracking on multiple frames
        * Repeat tracking on current frame…
    - **Measurement: Settings for adding and managing custom measurements**
        * Set measurements
        * Add custom measurement
        * Add combined measurement
    - Settings: Settings for changing the behaviour of tools, including **warning behaviour** and **not disabling tools after usage**
    - **Mode: change the mode**
        * Segmentation and Tracking, Cell cycle analysis, Viewer, Custom annotations

2.6 Create Data Structure with ImageJ/Fiji macros

Follow this steps to create the required data structure from the raw microscopy files using the provided Fiji macros:

1. If you don't have it, **install ImageJ/Fiji** from here Fiji Downloads.
2. **Download the Fiji macros** from here Cell-ACDC Fiji macros.
3. **Open the macro** with Fiji:

   If you have one or more microscopy files one for each position use the macro called `multiple_files.ijm`, while if you have multiple positions within a single file use the `single_file.ijm` macro.
4. **Modify the list of channels:**

In the first lines of the macro you will find the variable `channels` defined as follows:

```
channels = newArray("phase_contr", "mCitrine");
```

Modify that list with the correct order of channels as they are found in your microscopy file(s).

5. **Run the macro** with the `Run --> Run` menu
6. If you are using `single_file.ijm` **select the microscopy file**, if not **select the folder containing the microscopy file(s)**.

If successful, at the end of the process you should have all the Position folders correctly generated and ready to be opened in Cell-ACDC.

If you are having issues, feel free to open an issue on our GitHub page.

Until next time!

2.7 How to contribute to Cell-ACDC

Contributions to Cell-ACDC are **always very welcome**! If you have questions about contributing feel free to open a new thread on our forum.

2.7.1 Development process

1. If this is the first time you contribute:
   - Go to our GitHub page and click the "fork" button to create your own copy of the project.
   - Open a terminal window. On Windows I recommend using the PowerShell 7
   - Clone the forked project to your local computer (remember to replace *your-username* in the link below):

     ```
     git clone https://github.com/your-username/Cell_ACDC.git
     ```

   - Navigate to the `Cell_ACDC` directory:

     ```
     cd Cell_ACDC
     ```

   - Add the upstream repository:

     ```
     git remote add upstream https://github.com/SchmollerLab/Cell_ACDC.git
     ```

   - Now, you have remote repositories named:
     - `upstream`, which refers to the original `Cell_ACDC` repository
     - `origin`, which refers to your personal fork
   - Install the cloned Cell-ACDC in developer mode (i.e. editable) in a virtual environment using `venv` or `conda`:
     - `venv` (more info here)

       ```
       # Navigate to a folder where you want to create the virtual env
       cd ~/.virtualenvs

       # Create a virtual env with the name you like, e.g., ``acdc-dev``
       ```

(continues on next page)

(continued from previous page)

```
python -m pip venv acdc-dev

# Activate the environment (on PowerShell replace ``source`` with dot ``.``)
source ~/.virtualenvs/acdc-dev/bin/activate

# Navigate to the cloned folder path (parent folder of ``cellacdc``)
cd <path_to_Cell_ACDC>

# Install Cell-ACDC in developer mode
pip install -e .
```

- conda (Anaconda, Miniconda, Miniforge etc.)

```
# Create a virtual env with the name you like, e.g., ``acdc-dev``
conda create -n acdc-dev python=3.10

# Activate the environment
conda activate acdc-dev

# Navigate to the cloned folder path (parent folder of ``cellacdc``)
cd <path_to_Cell_ACDC>

# Install Cell-ACDC in developer mode
pip install -e .
```

2. Develop your contribution:
   - Navigate to the cloned folder path (parent folder of `cellacdc`):

```
cd <path_to_Cell_ACDC>
```

   - Pull the latest changes from upstream:

```
git checkout main
git pull upstream main
```

   - Create a branch with the name you prefer, such as 'new-segm-model':

```
git checkout -b new-segm-model
```

   - Commit locally as you progress (with `git add` and `git commit`). Please write good commit messages.
3. Submit your contribution through a Pull Request (PR):
   - Push your changes back to your fork on GitHub:

```
git push origin new-segm-model
```

   - Go to GitHub. The new branch will show up with a green "pull request" button – click it.

Note that if you want to modify the code of the Pull Request, you can simply commit and push to the same branch. GitHub will automatically add to the open Pull Request.

2.8 GUI tools

Overview of all tools available in the GUI.

Contents

2.8.1 File control

- **New File ("Ctrl+N"):** Create a new empty segmentation file.
- **Load Folder ("Ctrl+O"):** Loads folder.
- **Load older version ( ):** Load an older saved or unsaved version of the `acdc_output.csv` file (table with annotations and measurements).
- **Save ("Ctrl+Alt+S"):** Save.
- **Quick save ("Ctrl+S"):** Save only segm. file.
- **Show in explorer ( ):** Opens explorer with currently loaded folder.
- **Undo ("Ctrl+Z"):** Undo the last action
- **Redo ("Ctrl+Y"):** Redo the last undone actions

2.8.2 View options

- **Find ID ("Ctrl+F"):** Find and highlight ID. Press "Esc" to clear highlighted object.
- **Open Slideshow ("Ctrl+W"):** Opens slideshow.
- **Skip forward to new object ("Page up"):** Skip forward to the frame where a new object appears.
- **Overlay channels ( ):** Right-click on the button to overlay additional channels. To overlay a different channel right-click on the colorbar on the left of the image. Use the colorbar ticks to adjust the selected channel's intensity. You can also adjust the opacity of the selected channel with the "Alpha <channel_name>" scrollbar below the image.

  Note: This button has a green background if you successfully 'loaded fluorescence data'
- **Activate points layer ( ):** Activate points layer tools. You can add as many points layers as you want and customise their appearance. This can be used to visualize points from a table or add points with clicking and then use these points for models like Segment Anything (input prompts).
- **Add contours from different segmentation file ( ):** Add contours layer from another segmentation file
- **Ruler ( ):** Draw a straight line and show its length. Length is displayed on the bottom-right corner.

2.8.3 Edit tools: Segmentation and tracking

- **Brush ("B"):**
  - Edit segmentation labels with a circular brush.
  - Increase brush size with "UP/DOWN" arrows on the keyboard.
  - **Default behaviour:**
    - Painting on the background will create a new label.
    - Edit an existing label by starting to paint on the label (brush cursor changes color when hovering an existing label).
    - Press "Shift" to force drawing a new object
    - Painting in default mode always draws UNDER existing labels.
  - **Power brush mode:**
    - Power brush: press "B" key twice quickly to force the brush to draw ABOVE existing labels. NOTE: If double-press is successful, then brush button turns red. The brush cursor is always white.
    - Power brush will draw a new object unless you keep "Ctrl" pressed. –> draw the ID you start the painting from.
  - **Manual ID mode:**
    - Toggle the manual ID mode with the "Auto-ID" checkbox on the top-right toolbar.
    - Enter the ID that you want to paint.
    - NOTE: use the power brush to draw ABOVE the existing labels.
- **Eraser ("X"):**
  - Erase segmentation labels with a circular eraser.
  - Increase eraser size with "UP/DOWN" arrows on the keyboard.
  - **Default behaviour:**
    - Starting to erase from the background (cursor is a red circle) will erase any labels you hover above.
    - Starting to erase from a specific label will erase only that label (cursor is a circle with the color of the label).
    - To enforce erasing all labels no matter where you start from double-press "X" key. If double-press is successfull, then eraser button is red and eraser cursor always red.
- **Curvature Tool ("C"):** Left-clicks for manual spline anchors, right button for drawing auto-contour.
- **Magic Wand ("W"):** Left-click for single selection or left-click and then drag for continous selection.
- **Copy lost contour ("V"):** Hover onto lost object contour –> right-click to copy the contour as a new object.
- **Magic labeller ("L"):** Draw a rectangular ROI around object(s) you want to segment. Draw with LEFT button to label with last used model. Draw with RIGHT button to choose a different segmentation model.
- **Segment ("R"):** Segment with last used model and last used parameters. If you never selected a segmentation model before, you will be asked to choose one and initialize its parameters.
- **Manual background ("G"):**
  - **How to use:**

1. Select object to copy its shape.
2. Place the new shape on the background close to the source object.
3. Left-click to set the background ROI of the selected object.

– Note: right-click on a background ROI to remove it.

– HELP: Use this function if you need to set the background level specific for each object. Cell-ACDC will save the metrics *amount*, *concentration* and *corrected_mean* where the background correction will be performed by subtracting the mean of the signal in the background ROI (for each object).

- **Delete everything outside segmented areas ( ):** Select a segmentation file and delete everything outside segmented area.
- **Hull contour ("K"):** Right-click on a cell to replace it with its hull contour. Use it to fill cracks and holes.
- **Fill holes ("F"):** Right-click on a cell to fill holes.
- **Move object mask ("P"):** Right-click drag and drop a labels to move it around.
- **Expand/Shrink object mask ("E"):** Leave mouse cursor on the label you want to expand/shrink and press arrow up/down on the keyboard to expand/shrink the mask.
- **Edit ID ("N"):** Manually change ID of a cell by right-clicking on cell.
- **Manual bud separation ("S"):** Separate mother-bud fused together or separate objects that have the same ID. Right-click to attempt automatic separation or Shift+right-click to skip automatic attemp and go straight to manual mode.
- **Merge IDs ("M"): Merge/fuse two objects together.**
  – Usage: right-click on one of the two objects, keep the button clicked and release on the second object to merge (drag-and-drop).
- **Select object masks to keep ("K"):** Select the objects to keep. Press "Enter" to confirm selection or "Esc" to clear the selection. After confirming, all the NON selected objects will be deleted. Right- or left-click on objects to keep.
- **Remove object from analysis ( ):** Annotate that a cell is removed from downstream analysis. `is_cell_excluded` set to `True` in `acdc_output.csv` table. Done by right-clicking.
- **Annotate cell as dead ("D"):** Annotate that a cell is dead. `is_cell_dead` set to `True` in `acdc_output.csv` table.
- **Add deletion ROI ( ):** Add resizable rectangle. Every ID touched by the rectangle will be automatically deleted. Moving and resizing the rectangle will restore deleted IDs if they are not touched by it anymore. To delete rectangle `right-click on it --> remove`.
- **Add poly-line deletion ROI ( ):**
  – **How to use**
    1. Activate the button.
    2. Left-click on the LEFT image to add a new anchor point.
    3. Add as many anchor points as needed and then close by clicking on starting anchor.
    4. Delete an anchor-point with right-click on it.
    5. Add a new anchor point on an existing segment with right-click on the segment.
  – Add custom poly-line deletion ROI. Every ID touched by the ROI will be automatically deleted.
  – Moving and reshaping the ROI will restore deleted IDs if they are not touched by it anymore.

  - To delete the ROI `right-click on it --> remove`.

- **Delete bordering objects ( ):** Remove segmented objects touching the border of the image.
- **Repeat tracking ("Shift+T"):** Repeat tracking on current frame. Tracking method can be changed in `Tracking --> Select real-time tracking algorithm`
- **Manual tracking ("T"):** Select ID to track and right-click on an object to assign that ID.
- **Reset last segmented frame ( ):** Reset last segmented frame to current one. NOTE: This will re-enable real-time tracking for all the future frames.
- **Segment range of frames ( ):** Segment a range of frames with the "Magic labeller" tool.

2.8.4 Edit tools: Cell cycle analysis

- **Assign mother to bud ("A"):** Press with right button on bud and release on mother (right-click drag-and-drop).
- **Annotate as unknown history ("U"):** Useful for cells appearing from outside of the field of view.
- **Automatically assign bud to mother ( ):** Automatically assign buds to mothers using YeastMate.
- **Manually edit cell cycle annotations table ("Ctrl+Shift+P"):** Manually edit cell cycle annotations table.
- **Re-initialize cell cycle annotations table ( ):** Re-initialize cell cycle annotations table from this frame onward. NOTE: This will erase all the already annotated future frames information (from the current session not the saved information).

2.9 Models for automatic segmentation and tracking

Contents

Cell-ACDC has several models which can be used for segmentation of your data, as well as tracking of objects. Using the Segmentation module, or directly in the GUI, these models can be accessed.

2.9.1 Preinstalled models

The two most important models are:

- YeaZ
- Cellpose

This does not mean that you should ignore the other models already included! Each model has its own strengths and weaknesses, so make sure to try out different models to find the right fit for you. Give the corresponding publications a read for further information:

- Segment Anything Model (SAM)
- StarDist
- YeastMate
- Omnipose
- Delta
- DeepSea
- TAPIR
- Bayesian tracker (btrack)
- Trackpy

YeaZ

For greater detail, please read the publication, or visit their GitHub page.

YeaZ is a **convolutional neural network**. As the name suggests, it was developed and trained on images of yeast. This means that it **works great with yeast**, but not so much with other things. YeaZ can be used for both **segmentation and tracking**.

However, YeaZ does not work well with bright field images, as it was not trained with such images. **YeaZ2** should have **improved performance for bright field**, so the second version should be a good solution when working with yeast.

Cellpose

For greater detail, please read the publication or their web page.

Cellpose was trained on a **more diverse** data set than YeaZ, and thus can be used for segmentation of a wider range of data. If YeaZ is not the right fit for you, Cellpose might be.

2.9.2 Adding a new Model

If none of the included models suits your purposes, adding your own model might be a good idea.

Adding a segmentation model

Adding a segmentation model in a few steps:

1. Create a **new folder** with the models's name (e.g., YeastMate) inside the `/cellacdc/models` folder.

Tip: If you **don't know where Cell-ACDC was installed**, open the main launcher and click on the `Help --> About Cell-ACDC` menu on the top menu bar.

---

2. Create a `__init__.py` file inside the model folder. In this file you can handle **automatic installation** of the module required by your model. For example, if your model requires the module tensorflow you can install it manually, or programmatically as follows:

```
import os
import sys

try:
    import tensorflow
    # Tries to import the module, in this case btrack

except ModuleNotFoundError:
    subprocess.check_call(
        [sys.executable, '-m', 'pip', 'install', 'tensorflow']
        # If the model is not found (so its not installed),
        # it will try installing it using btrack
    )
```

   Add any line of code needed to initialize correct import of the model.

3. Create a new file called `acdcSegment.py` in the model folder with the following template code:

```
import module1 #here put the modules that the tracker needs
import module2

class Model:
    def __init__(self, **init_kwargs):
        script_path = os.path.dirname(os.path.realpath(__file__))
        weights_path = os.path.join(script_path, 'model', 'weights')

        self.model = MyModel(
            weights_path, **init_kwargs
        )

    def segment(self, image, **segment_kwargs):
        lab = self.model.eval(image, **segment_kwargs)
        return lab
```

The **model parameters** will be **automatically inferred from the class you created** in the `acdcSegment.py` file, and a widget with those parameters will pop-up. In this widget you can set the model parameters (or press Ok without changing anything if you want to go with default parameters).

Have a loot at the `/cellacdc/models` folder here for **examples**. You can for example see the `__init__.py` file here and the `acdcSegment.py` file here for YeaZ2.

Adding a tracker

This only takes a few minutes:

1. Create a **new folder** with the trackers's name (e.g., YeaZ) inside the `/cellacdc/trackers` folder.

Tip: If you **don't know where Cell-ACDC was installed**, open the main launcher and click on the `Help --> About Cell-ACDC` menu on the top menu bar.

---

2. Create a `__init__.py` file inside the folder. In this file you can handle **automatic installation** of the module required by your tracker. For example, if your tracker requires the module `btrack` you can install it manually, or programmatically as follows:

```
import os
import sys

try:
    import btrack
    # Tries to import the module, in this case btrack

except ModuleNotFoundError:
    subprocess.check_call(
        [sys.executable, '-m', 'pip', 'install', 'btrack']
        # If the model is not found (so its not installed),
        # it will try installing it using btrack
    )
```

   Add any line of code needed to initialize correct import of the model.

3. Create a new file called `trackerName_tracker.py` (e.g.,`` YeaZ_tracker.py``) in the tracker folder with the following template code:

```
import module1 #here put the modules that the tracker needs
import module2

class tracker:
    def __init__(self):
        '''here put the code to initialize tracker'''

    def track(self, segm_video, signals=None, export_to=None):
        '''here put the code to that from a
           segmented video Returns the tracked video
        '''
        return tracked_video
```

Have a look at the already implemented trackers. The `cellacdc/trackers` folder can be found here, with YeaZ_tracker.py as an example for YeaZ.

That's it. Next time you launch the segmentation module you will be able to select your new tracker.

2.10 Scripts to correct shifts in bidirectional scanning

Using a bidirectional microscope can cause shifts in every second x-axis, resulting in extremely bad segmentation and tracking performance. These scripts correct this.

Work-flow

Run this **before** dataprep but after creating a structure out of microscopy file.

If you have multiple positions

1. activate the `acdc` environment
2. Navigate to `cd \Cell_ACDC\cellacdc\scripts` (if you have installed Cell_ACDC in a different directory navigate to that directory first)
3. Use `python correct_shift_X_multi.py [path to folder] [initial shift]` to run the script. Path to folder would be for example `C:\Users\SchmollerLab\Documents\MyData\TimelapseExp\2022-08-04` when one of the .tif files is in `C:\Users\SchmollerLab\Documents\MyData\TimelapseExp\2022-08-04\Position_8\Images`
4. The program will run, pay attention to the console for user input requests!

Note: After the program is finished, you can run the data prep module. Check before running it that all frames are fine now, as sometimes the sign of the shift changes for a few frames. In this case, you can use the `correct_shift_X_single.py` script. Sometimes you need to use a negative shift! Most of the times (but not always) the shift is the same for all positions.

The expected structure of the folder is as the "Create data structure" module from Cell_ACDC creates.

If you have a single position

As above, except in step 3. You need to run `python correct_shift_X.py [path to tif file] [initial shift]`

Note: Notes As above

If you need to change single frames

As above, except in step 3. You need to run `python correct_shift_X_single.py [path to tif file] [initial shift] [start frame] [end frame]`

Note: Notes As above

Usually, the sign of the shift changes in these frames. Since you usually should have run a shift on the TIF file before, now you need to shift it with twice the amount back. If the original shift was 3, now you need to apply a shift of -6!

The numbers for the frames match exactly with the numbers shown in the data prep module, so the first frame is the first and not the zeroth.

Possible problems with solutions

- I get errors from 'imshow': Change `PREVIEW_Z_STACK` and `PREVIEW_Z` (see Configs) in the script to values that make sense for you. Worst case try 0 for both.
- TIF files are not found: Change `INCLUDE_PATTERN_TIF_SEARCH` for additional TIF files and `INCLUDE_PATTERN_TIF_BASESEARCH` if `_multi` does not find any paths accordingly.

- Wrong TIF files found: Refer to TIF files are not found.
- The shift is different depending on where on one picture I'm currently at: No fix, usually this effect is low enough to not cause problems with segmentation and tracking. *pew*

Configs

There are quite a few things you can change in scripts. To change them, change them in configs.json in the `scripts` folder. For the regex expression please find `regex.txt`

Same in all scripts

`NEW_PATH_SUF`: Changes the suffix of the new files. Leaving it empty causes old files to be overwritten, which is recommended, as otherwise the data prep process will also align the old files.

`PREVIEW_Z_STACK`: Changes the frame which is used in the preview.

`PREVIEW_Z`: Changes the z-slice which is used in preview.

`INCLUDE_PATTERN_TIF_SEARCH`: Regex expression which is used to filter the .tif files if you choose to search for other TIF files. If you don't know regex, ask Chat_GPT to generate one for you by giving it examples of file names and then asking it to generate a regex code which excludes the files you want to exclude. In the code it is used in a `re.match` function which iterates over all TIF files in the folder.

_multi

`PRESET_SHIFT`: Allows you to set a standard shift.

`INCLUDE_PATTERN_TIF_BASESEARCH`: Same as in `INCLUDE_PATTERN_TIF_SEARCH`, but this regex expression is used to match TIF files which are used to determine the shift.

`FOLDER_FILTER`: Filter which is applied in order to make sure only folders which contain tif files (and not for example the original microscopy file) are considered. The first function `finding_base_tif_files_path()` is used to find the base TIF files. If your basic directory structure is different, change this function accordingly. `root_path` is simply the argument parsed from user input. `base_tif_files_paths` is the path directly to your main TIF file which should be used to determine the shift.

`tif_files_paths` is the path to the folder in which they are contained.

2.11 Cell-ACDC output data

Files saved by Cell-ACDC for a **fully analysed experiment** (example with original raw microscopy file called `Example1` and first position `_s01_`)

File:	Description:
`Example1_s01_acdc_output.csv`	Main table containing cell cycle annotations and additional metrics such as mean, median etc. for all the loaded channels plus all the region properties (as calculated by skimage.measure.regionprops) for each segmented object.
`Example1_s01_act1.tif`	.tif file for the channel called `act1`.
`Example1_s01_cdc10.tif`	.tif file for the channel called `cdc10`.
`Example1_s01_last_tracked_i.txt`	**Last visited frame** in "Segmentation and Tracking mode" with the main GUI.
`Example1_s01_metadata.csv`	Table containing the **metadata** such as number of frames, number of z-slices, pixel size etc.
`Example1_s01_phase_contr.tif`	.tif file for the channel called `phase_contr`.
`Example1_s01_segm.npz`	**Segmentation labels**. This is a numpy array (compressed).
`Example1_s01_segmInfo.csv`	Table containing information such as which z-slice or z-projection was used for segmentation or saving metrics of each channel.
`Example1_s01_align_shift.npy`	Numpy array containing the **shifts** applied to each frame when **aligning**. Useful for reverting to non-aligned state.
`Example1_s01_dataPrepROIs_coords.csv`	Table containing the **coordinates** of the **ROI** that was used to either **crop**, or **segment** only objects in the ROI. This is created during the data prep or segmentation stage.
`Example1_s01_phase_contr_aligned.npz`	Aligned data for the channel called `phase_contr`. This is a numpy array (compressed).
`Example1_s01_phase_contr_aligned_bkgrRoiData.npz`	**Data** from the **background ROIs** generated at the data prep stage.
`Example1_s01_phase_contr_dataPrep_bkgrROIs.json`	**Coordinates** of the **background ROIs** generated at the data prep stage.

2.12 Troubleshooting

List of issues:

2.12.1 Import cv2 failed: `ImportError: DLL load failed`

If you see the following error:

```
ImportError: DLL load failed: The specified module could not be found.
```

it is very likely that you are on a Windows 10/11 `N` or `KN` version.

You can check if you have an `N` or `KN` version by typing `winver` in the Windows search bar and clicking on `winver` as in this screenshot:

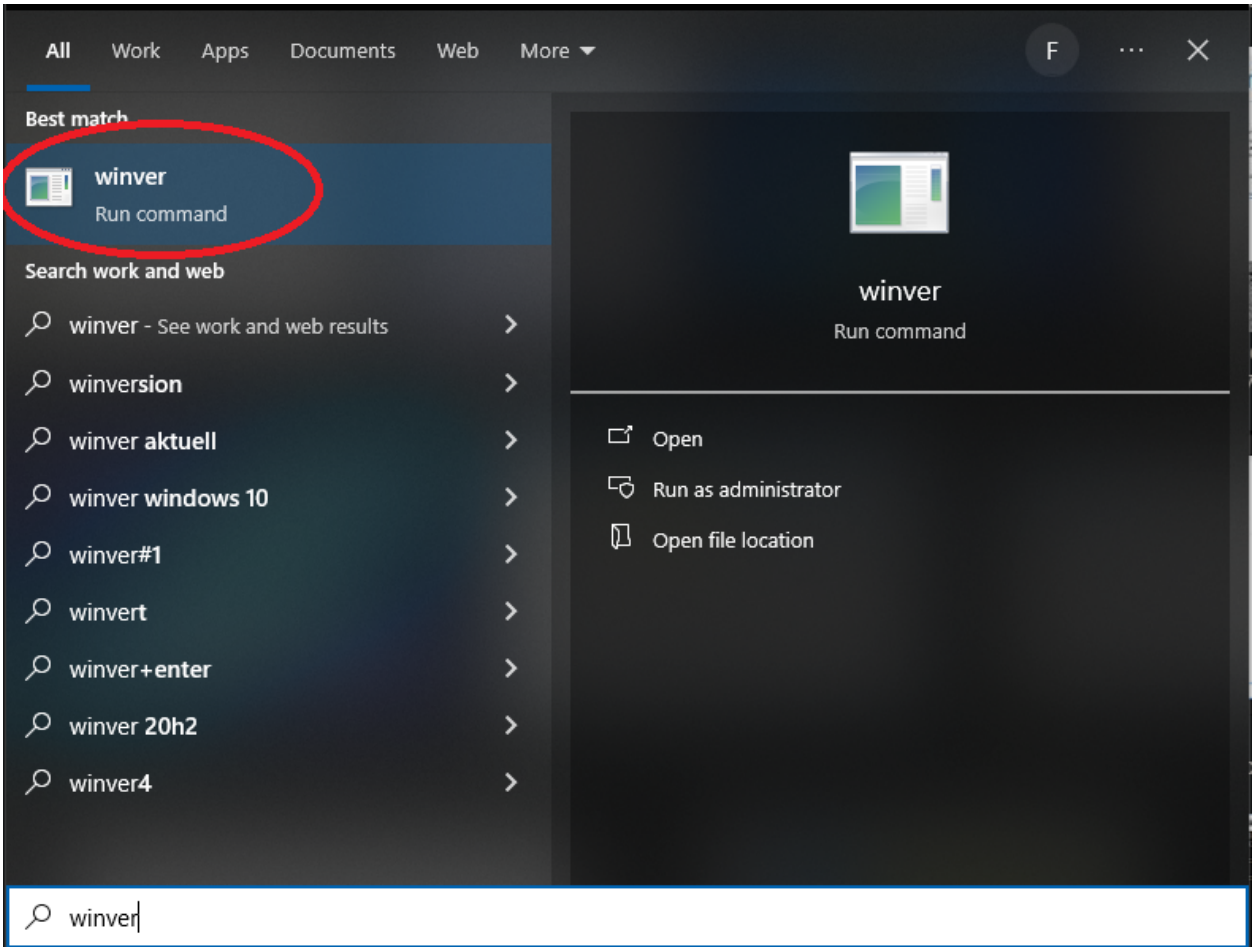


Fig. 8: Runnning `winver` from Windows search bar

You will be able to see the Windows version in the window that opens:

If you see a `N` or `KN` in the Edition field then you need to **install the Media Feature Pack** which is required by `opencv` and it is not installed on `N` and `KN` versions of Windows.

To install the Media Feature pack type `optional feature` in the Windows search bar and you should see something like "Add an optional feature". Click on that and then click on "Add a feature" on the top. Search for `Media Feature`

Fig. 9: Windows version

Pack and install it.

After installation, re-start the computer and try Cell-ACDC again.

If you don't manage to solve your issue, feel free to contact us by opening a new issue on our GitHub page or by sending me an email at .

Until next time!

2.13 Versions

This should not be considered a complete list of versions

Contents

2.13.1 Update v1.2.4

First release that is finally available on PyPi.

Main new feature: custom trackers! You can now add any tracker you want by implementing a simple tracker class. See the manual at the section "**Adding trackers to the pipeline**".

Additionally, this release includes many UI/UX improvements such as color and style customisation, alongside a inverted LUTs.

2.14 Resources

Contents

2.14.1 Resources

- *Installation guide*.
- User Manual with **detailed instructions**.
- Publication of Cell-ACDC.
- Forum for discussions (feel free to **ask any question**).
- **Report issues, request a feature or ask questions** by opening a new issue here.
- X thread.

2.14.2 Usage

For details about how to use Cell-ACDC please read the User Manual downloadable from here

CHAPTER

THREE

IMPORTANT LINKS

- GitHub
- Publication
- Forum