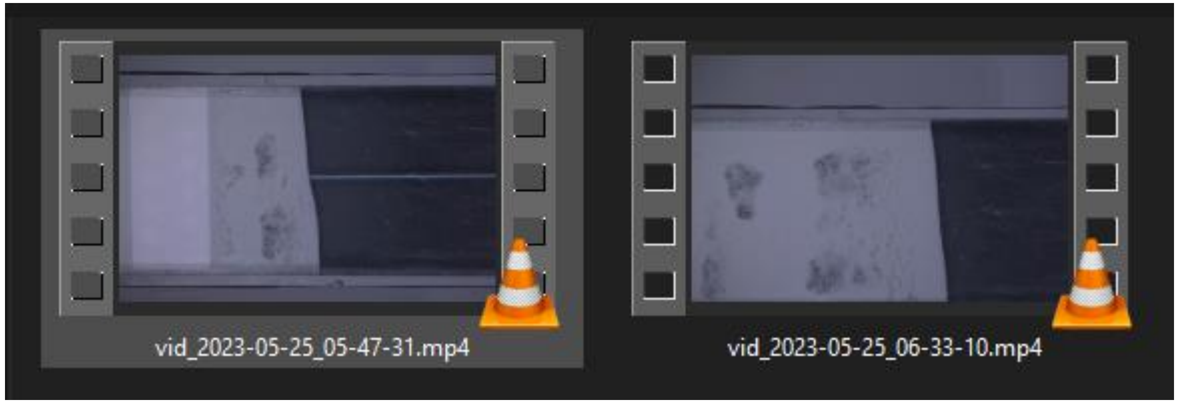
# Data Analysis – Session 1

08.06.2023

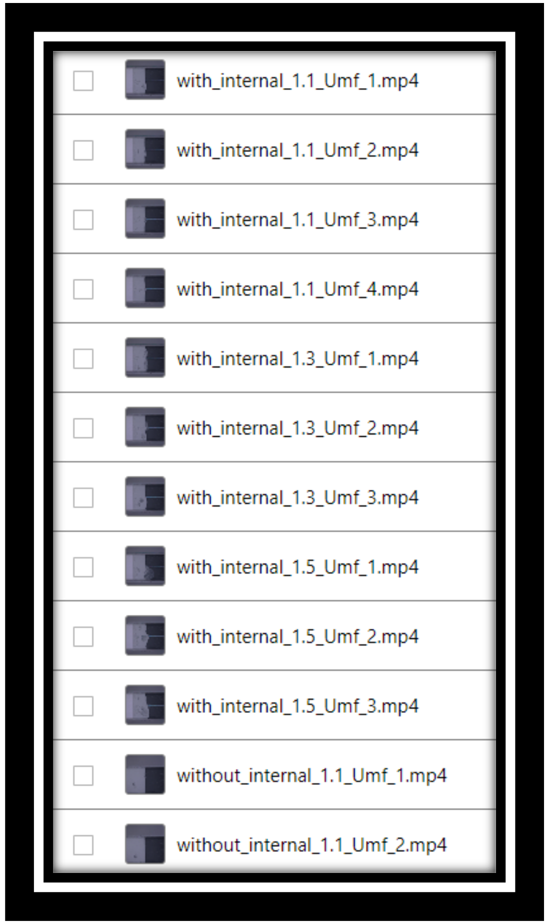
# Step 1) File renaming

- Give the files meaningful names to simplify evaluation



vid_2023-05-25_05-47-31.mp4

vid_2023-05-25_06-33-10.mp4



□ with_internal_1.1_Umf_1.mp4

□ with_internal_1.1_Umf_2.mp4

□ with_internal_1.1_Umf_3.mp4

□ with_internal_1.1_Umf_4.mp4

□ with_internal_1.3_Umf_1.mp4

□ with_internal_1.3_Umf_2.mp4

□ with_internal_1.3_Umf_3.mp4

□ with_internal_1.5_Umf_1.mp4

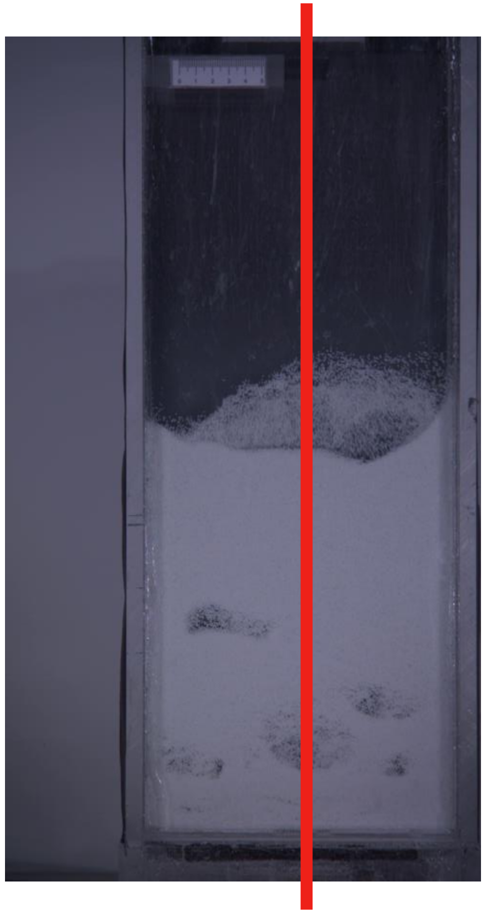
□ with_internal_1.5_Umf_2.mp4

□ with_internal_1.5_Umf_3.mp4

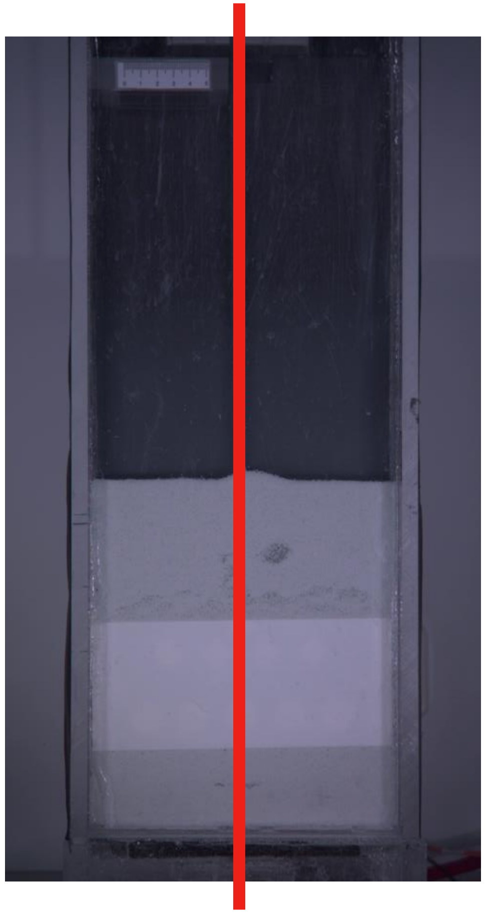
□ without_internal_1.1_Umf_1.mp4

□ without_internal_1.1_Umf_2.mp4

# Step 2) Crop and align



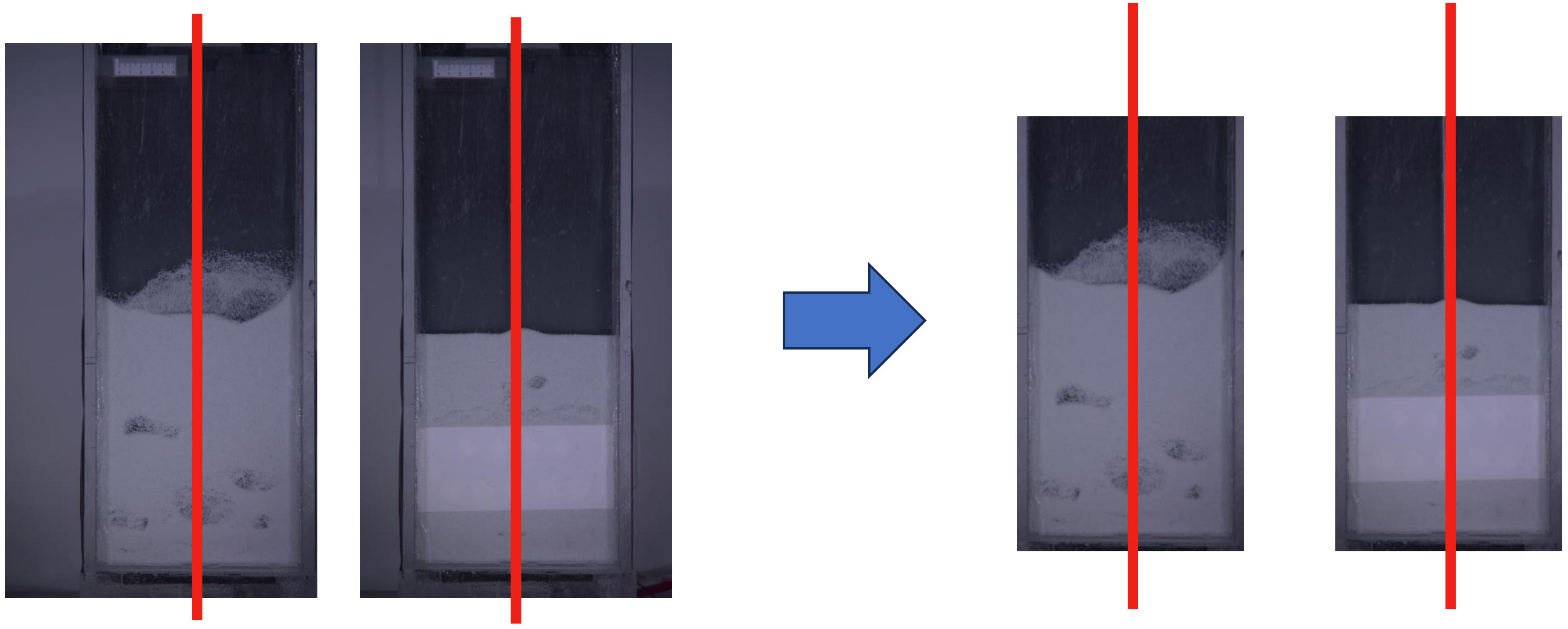
without_internal_1.5_Umf_3.mp4



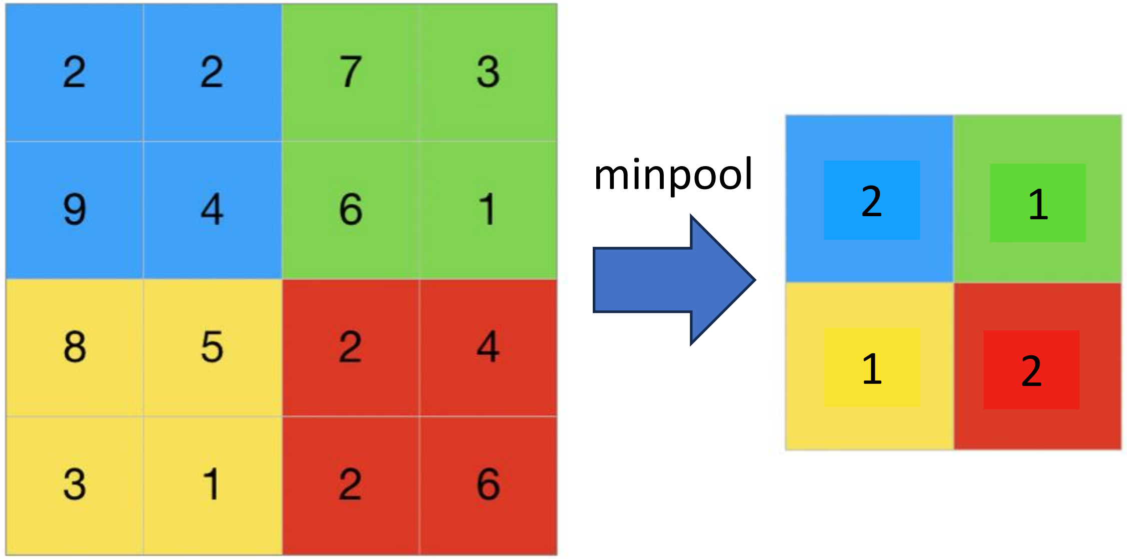
with_internal_1.1_Umf_1.mp4

- Both measurements are from the same group, but their alignment is different
- Use tool / fix point to give every measurement the same alignment

# Step 2) Crop and align



# Reduce resolution with minpool



|   |   |   |   |
|---|---|---|---|
| 2 | 2 | 7 | 3 |
| 9 | 4 | 6 | 1 |
| 8 | 5 | 2 | 4 |
| 3 | 1 | 2 | 6 |

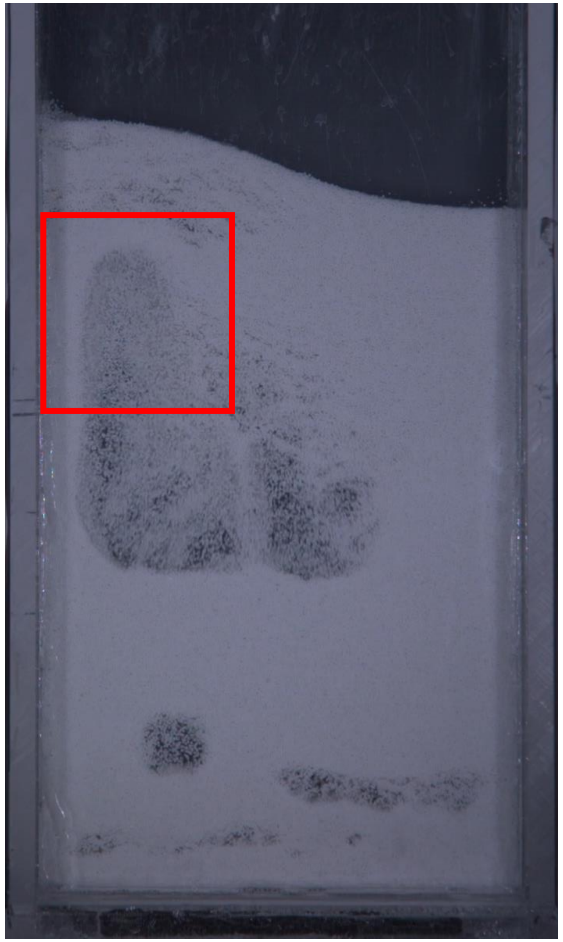
minpool

|   |   |
|---|---|
| 2 | 1 |
| 1 | 2 |

- Minpool reduces a 2x2 pixel grid to one pixel. The resulting pixel has the minimum value of the previous 4 pixels.
- By performing the operation, the resolution get’s halved
  Example: 1000x1000 -> 500x500

# Reduce resolution with minpool

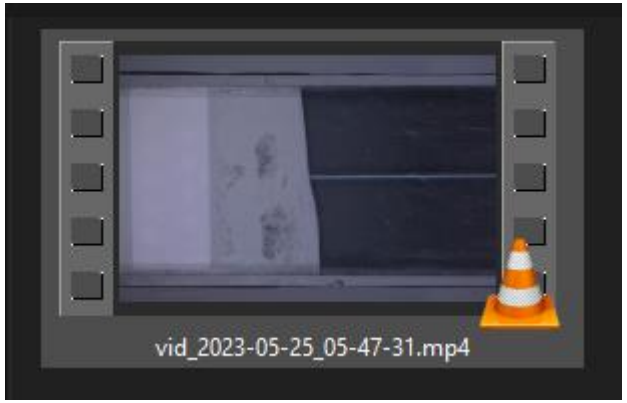
- Reasoning: Some bubbles are hardly visible. Only a few darker pixels indicate their existence.
- Normal down sampling would average pixel values, darker values would vanish.



# Reduce number of frames

- You have recorded 1000 frames for each measurement
- To reduce file size and processing time, 100 frames of each measurement have been chosen for processing

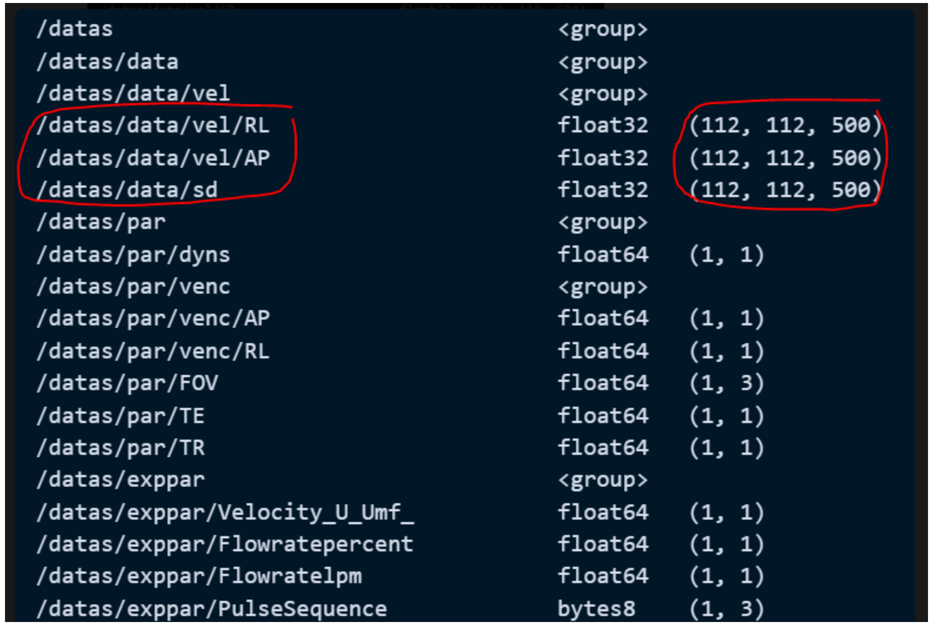
# Save as h5 file



- File format from camera: MP4
  MP4 is a format for compressed video
    - Small filesize
    - Data cannot be accessed directly

Render mp4 video into a 3D tensor (height * width * frames)
and save into a h5 file

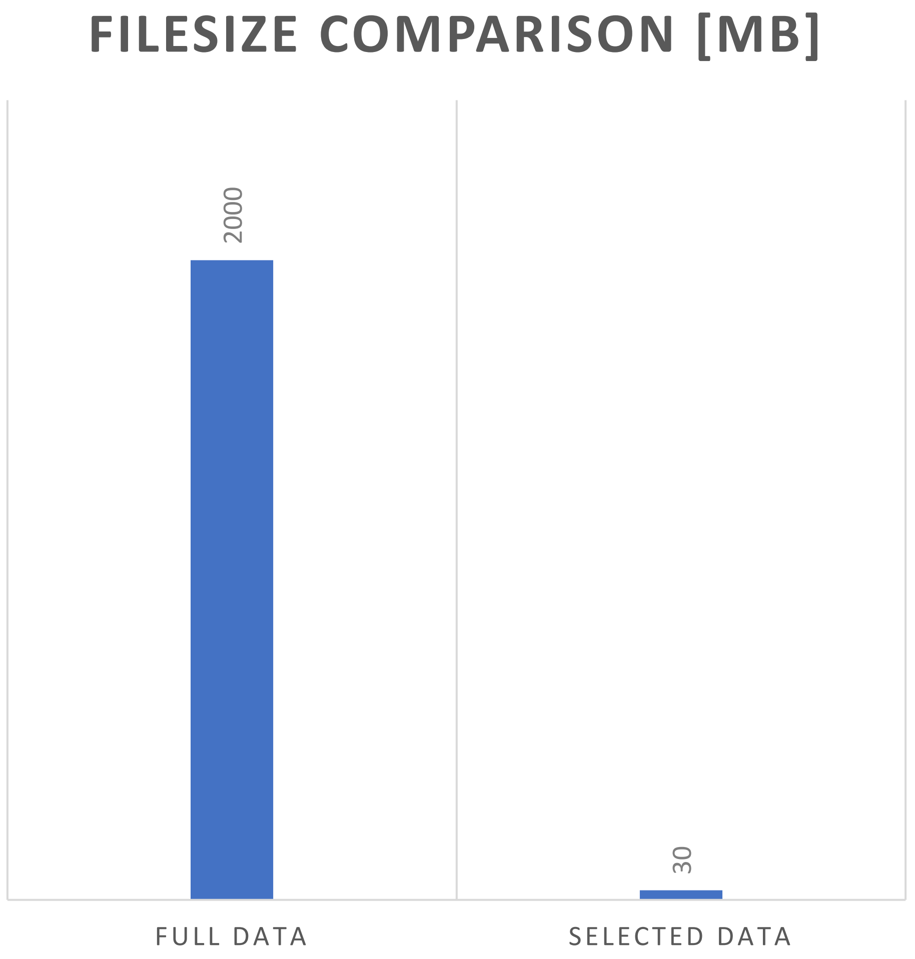
- New file format: H5
  H5 is a container format to store variables
    - Uncompressed
    - Data can be organized by groups and keys



```
/datas                              <group>
/datas/data                         <group>
/datas/data/vel                     <group>
/datas/data/vel/RL                  float32   (112, 112, 500)
/datas/data/vel/AP                  float32   (112, 112, 500)
/datas/data/sd                      float32   (112, 112, 500)
/datas/par                          <group>
/datas/par/dyns                     float64   (1, 1)
/datas/par/venc                     <group>
/datas/par/venc/AP                  float64   (1, 1)
/datas/par/venc/RL                  float64   (1, 1)
/datas/par/FOV                      float64   (1, 3)
/datas/par/TE                       float64   (1, 1)
/datas/par/TR                       float64   (1, 1)
/datas/exppar                       <group>
/datas/exppar/Velocity_U_Umf_       float64   (1, 1)
/datas/exppar/Flowratepercent       float64   (1, 1)
/datas/exppar/Flowratelpm           float64   (1, 1)
/datas/exppar/PulseSequence         bytes8    (1, 3)
```

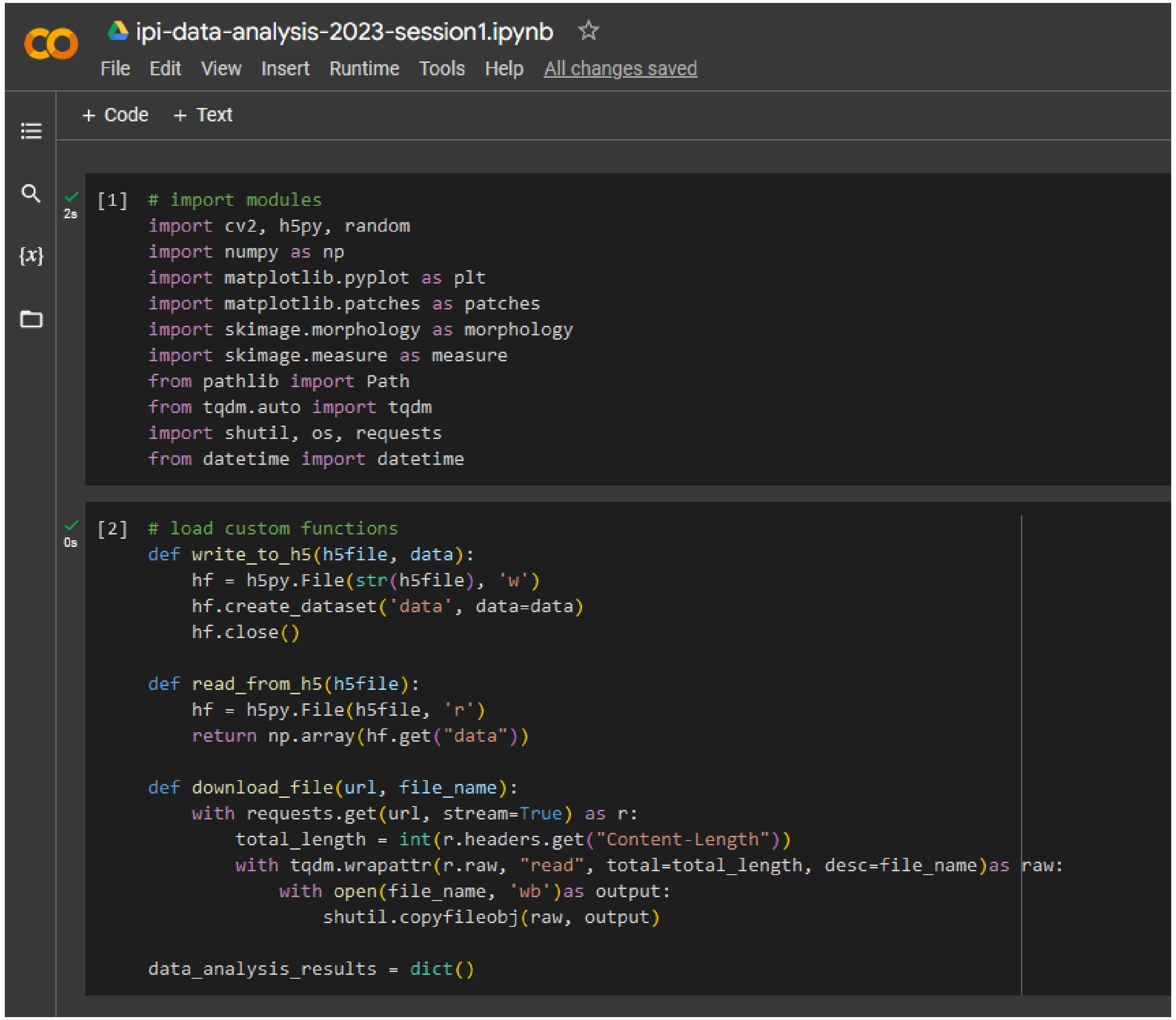
# File size optimization

- With the 1920x1080 resolution and 1000 frames, even with number type uint8 we end up with 2Gb for each recording
- $1920 \times 1080 \times 1000 \times 1\text{byte} = 2.073.600.000\text{bytes}$
- With all the optimizations, we end up with a ~30 Mb file



# Google Colab

- Run Python interactively, step by step
- Run Python on a website without installing it on your computer



```python
# import modules
import cv2, h5py, random
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import skimage.morphology as morphology
import skimage.measure as measure
from pathlib import Path
from tqdm.auto import tqdm
import shutil, os, requests
from datetime import datetime
```

```python
# load custom functions
def write_to_h5(h5file, data):
    hf = h5py.File(str(h5file), 'w')
    hf.create_dataset('data', data=data)
    hf.close()

def read_from_h5(h5file):
    hf = h5py.File(h5file, 'r')
    return np.array(hf.get("data"))

def download_file(url, file_name):
    with requests.get(url, stream=True) as r:
        total_length = int(r.headers.get("Content-Length"))
        with tqdm.wrapattr(r.raw, "read", total=total_length, desc=file_name)as raw:
            with open(file_name, 'wb')as output:
                shutil.copyfileobj(raw, output)

data_analysis_results = dict()
```

# Further resources

- https://www.ipi.tuhh.de/process-imaging/