Temporal Synchronization

What is temporal synchronization?

- Aligning several video streams such that it is frame-accurate
- This then allows to extract frames which are the basis / inputs for computer vision models
- Aligning means fitting a linear function y = mx + b where x is the frame index, y the aligned timestamp, m incorporates the temporal drift and b the temporal offset
- The offset b is due to the shifted start of the recording of each camera
- The temporal drift m is due to slight variations in the internal clocks of the cameras (while the drift itself is very small, it's still necessary to account for it to have a frame-accurate synchronization)

How is the fitting performed (what's happening in the code)?

- To get the x and y values the RocSync is used (ideally shown at the start and end of a recording)
- RocSync: ARuCo marker in the center, 4 LEDs at the corners, 100 LEDs in a circle (100ms), counter of these revolutions in binary as a line above the ARuCo marker. Additionally, infrared sources for infrared cameras (Atracsys)
- Basic principle: Each frame is analyzed w.r.t. whether the ARuCo marker is seen or not. If it's seen then the 4 LEDs are used to translate and rotate the RocSync such that it's positioned head-on. Then the binary number of revolutions is read off to get the actual timestamp. Moreover, the circle segment of LEDs is used to get the exposure time. The actual timestamp is then our y and the frame number that led to this timestamp is our x.
- After a RANSAC regressor is used to perform a fit on the data and from this m and b can be deduced.

How is the fitting performed (as a user)?

- To save some time (the algorithm doesn't need to check each frame of a whole recording), it's better to first analyze all recordings to check when the RocSync is shown. Usually these are two time windows. Since the time windows are equal for all cameras of one recording, the time windows need to be large enough such that the RocSync is seen on each camera.
- Identify cameras that have low brightness and write name fragments of these cameras in the main.py file in line 357 such that the brightness is increased
- After this pre-processing step, the following command line prompt is used: **rocsync path/to/data --start1 hh:mm:ss --end1 hh:mm:ss --start2 hh:mm:ss --end2 hh:mm:ss -o path/to/output.json**
- In practice this command looks like this: **rocsync F:/visceral_v1 --start1 00:00:00 --end1 00:15:00 --start2 01:00:00 --end2 01:15:00 -o F:/visceral_v1/calibration/time_synchronization_RecX.json**

- Make sure to use the newest commit of Jaro's code directly in the RocSync repository and not the forked version in the calibration repo of the BAL-ROCS-BUT-COOL organization.

Structure of the output .json file (dictionary) only most important explanations:
Key – location of camera X
Value – {"n_frames": number of frames of recording
"n_considered_frames" and "n_rejected_frames": frames that were considered for the fit vs frames that were outliers
"measured_fps": self-explanatory
"speed_factor": gives a relative feeling of how strong the drift of a certain camera is
"first_frame": temporal offset b in ms
"last_frame": in ms, used with "first_frame" and "n_frames" to calculate the temporal drift m via (last_frame-first_frame)/n_frames}

How to check the synchronization?

- Usually there should be two light switches visible in a recording
- Choose one camera and identify the two time windows of around 5-10 seconds during which the light switch happens (if possible choose a light switch of the whole room rather than only the OR lamp (see below for details))
- Use the extract_clips_as_png.py script in the calibration repository to extracted synchronized frames:
    - Line 27: change the name of the synchronization file (ToDo -> implement such that it can be passed via command line argument)
    - Create a file clips_config.json that looks like this:

```
[
    {
        "start": "00:05:15.000",
        "end": "00:05:25.000"
    },
    {
        "start": "01:06:45.000",
        "end": "01:06:55.000"
    }
]
```

    - Via the command line input the following prompt: **python further_scripts/extract_clips_as_png.py --dataset_folder path/to/dataset_folder --clips_to_extract_json path/to/clips_config.json --target_fps 30 --from_raw_camera_time_of_camera camera_used_for_clips_config**
    - This gives a new folder called "synced_clips_png" in the location of the dataset folder
- Navigate to the "synced_clips_png" folder and go to the camera folder of the one used for the clips_config.json. Identify two frames between which the light

switch happens. This step is not straightforward with the current datasets, since the light switch is gradual and not immediate. Therefore, there might be some faint light already for some cameras while for others it is still dark. For the room light -> choose the frames where the room has no dark patches anymore. For the OR lamp -> Try to identify the strongest brightness increase.
- Check all cameras if for the given frames the same change in light is seen (if it's more than one frame off, it's clear something went wrong; if it's only one frame off, look closely as it might be not due to the synchronization but the light source itself and therefore it could be that there is no issue)

Common Issues:
- Near field cameras are usually low in brightness -> adjusting brightness might help
- If the ChAruCo board and the RocSync are visible at the same time the wrong AruCo marker might be chosen (but this should be fixed in the newest commit)
- If cameras are zoomed in too far the RocSync might not be visible completely
- Binary clock is not readable
- The RocSync is only shown for few frames or from too far away
- The cameras were shut off too early (then simply choose another moment to compare these cameras, usually egocentric cameras and Aria glasses)
- The cameras shut off early due to empty battery
- Due to some (yet) unknown reasons, the synchronization for the Canon cameras and ZedMini seems harder

Manual synchronization procedure:
- In case where there is no solution for the above issues, it's easiest to perform a manual synchronization and look at the cause to the issue later
- For this, two correspondences need to be found (one at the start and one at the end of the video). The first correspondence is usually the first light switch already used to check the synchronization. The second one is ideally also a light switch. In the case of a camera running out of battery or an egocentric camera, some other correspondence is needed (fast movement, audio, etc.)
- Use the Manual_Temp_Synch.py script to perform the manual synchronization:
    - Create a new time_synchronization.json file where only the cameras are included that are frame-accurately synched.
    - folder1 and folder2 names correspond to the two folders of the temporal synchronization check in "synched_clips_png".
    - Change the name of the synchronization json file for the cam_info variable
    - From the temporal synchronization check you already know the frame number where the transition of the light brightness happened, i.e., light switch happens between x1 and x2. For both light switches, remember x1 and change it for the y_values_switch variables as x_user.

- Change the camera_name to the desired one that should be manually synched.
- Change the folder name and the time windows in the detect_light_switch_frames function during which the correspondence happens. If the correspondence is a light switch, it usually works with the given code. If the correspondence is not a light switch, make sure, that the correspondence happens in the middle of your time window and you need to input light_switch = False. It may be necessary in that case to increase the number of pre and post frames extracted from the middle of the time window to see the correspondence.
- Run the script. It will output two folders with extracted frames. Ideally the desired correspondence is seen in these frames. If not, change the number of frames extracted before and after the increase in brightness happens or of the other correspondence.
- Now you have to match the frame of the new folder with the one from the synchronization check. Note down the frame number of the camera for which the manual synchronization is performed (x1_cam and x2_cam).
- Run the script again. Now the script extracts all actual timestamps for all cameras that are synched perfectly (y's) and the x's are x1_cam and x2_cam. Then a robust regression is performed to get the new fit.
- Now you can add a new entry to the time_synchronization.json file where especially the speed_factor, first_frame, n_frames, measured_fps, last_frame is needed.
- To check if you did the manual synchronization correctly, perform the extraction of frames again. If it's not yet fully synched, slightly shift x1_cam or x2_cam respectively, perform the fit again, and check again.