User Guide

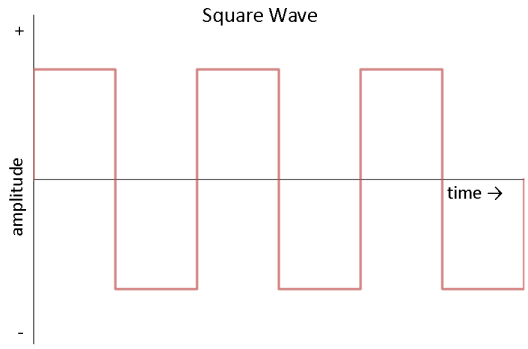
The set up

Cameras

The two cameras are set in trigger mode: they only acquire a frame when they receive a 5v TTL pulse. This ensures a more consistent frame rate and that the **two cameras acquire frames in a synchronised manner**.

The trigger is generate by an arduino (see section on arduinos below), and is shaped as a square wave:

The rising edge of the square wave is what trigger frame acquisition. **The period of the wave determines the frame rate**.



A square wave

Arduinos

The set up currently needs two arduinos:

- **Trigger arduino** → used to produce the square wave triggers for the cameras
- **Sensors arduino** → used to register the state of the force sensors.

Trigger arduino

The trigger arduino is used to send the triggers for frame acquisition. The period of the square wave is what sets the framerate of the recording.

To change the framerate, you'll need to modify the *`wait time`* in the *camera_trigger.ino* script that controls the trigger arduino. This can be found here: *"APA_setup\Arduino\camera_trigger\camera_trigger.ino"*. This is the content of that script:

```
// specify the pins of the camera triggers outputs
int pinCam1 = 13;
int pinCam2 = 10;

void setup() {
  pinMode(pinCam1, OUTPUT);
  pinMode(pinCam2, OUTPUT);

}

void loop() {
  digitalWrite(pinCam1, HIGH);
  digitalWrite(pinCam2, HIGH);
  delay(2);  // <- change this to modify framerate

  digitalWrite(pinCam1, LOW);
  digitalWrite(pinCam2, LOW);
  delay(3); // <- change this to modify framerate
}
```

To change framerate you'll nee to change the delay(n) lines, the sum of the two delays should be equal to the duration in milliseconds of one frame, given by **1000/fps = frame_duration**.

Sensors arduino

This is the arduino that records the state of the sensors every time a frame is acquired. This is done by running firmata_ on it, an arduino script that allows for control of the arduino through serial, i.e. through python.

To control the arduino via python, the APA_setup code uses pyFirmata_ which allows the user to read the state of the analog input channels of the arduino which are the ones that receive the analog signal from the amplifier that in turn receives the DC→AC converted voltage of the force sensors.

The amplifier's output range is between 0 and 5V, the analog inputs on the arduino also read in the 0-5V range, but pyFirmata converts this into a 0-1 range. So the state of a force sensor at any given time, as read through pyFirmata will be a float number between 0 (no force) and 1 (max force).

More details about how the sensors states are read in python in the Code section below.

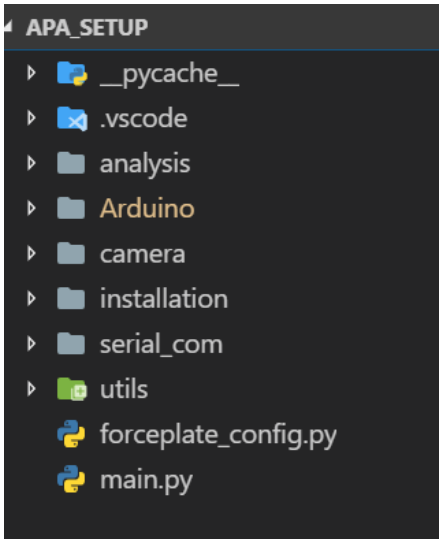
The code

Folder structure

The code is on a GitHub repository called APA_setup which can be found here:

Egzona66/APA_setup
https://github.com/Egzona66/APA_setup

Inside APA_setup are a number of subfolders and scripts.



The **folders** are organised thematically:

- analysis: contains scripts that are used to analyse data, plot stuff, create videos with plots...
- Arduino: contains the ".ino" scripts used to manage the arduinos
- camera: contains *camera.py* the main script that takes care of video acquisition
- installation: contains details that make it easier to set up the same code on a new computer
- serial_com: contains scripts that are used for serial communication with the sensors arduino through pyFirmata
- utils: has scripts that contain a collection of useful functions that are used through the analysis

The two main scripts used to run an experiment are **forceplate_config** and **main**. More details in the Using the code section below.

Using the code

Starting and Stopping an experiment

Starting an experiment requires two steps:

1. Checking the experiment configuration in forceplate_config
2. Running the experiment by running main

Forceplate_config holds the configuration options for both running and analysing experiments. Which options are important for which process are clearly marked. These are the options for running a recording:

```
"""
    ############## EXPERIMENT CONFIG  ####################
"""
# ! Change these for every recording
experiment_folder = "E:\\Egzona\\test"   # ? This should be changed for everyexperiment to avoid overwriting
experiment_name = "tes22t"  # should be something like YYMMDD_MOUSEID, all files for an experiment will start with this name
experiment_duration = 5*60  # acquisition duration in seconds, alternatively set as None

# * Live video frames display and sensors data plotting
live_display = False  # show the video frames as video is acquired
live_plotting = False

# * Check that these options are correct
com_port = "COM5"  # port of the arduino running Firmata for data acquisition
acquisition_framerate = 200  # fps of camera triggering -> NEED TO SPECIFY SLEEP TIME IN ARDUINO for frame triggering


overwrite_files = False # ! ATTENTION: this is useful for debug but could lead to overwriting experimental data
# So this number is just indicative but the true acquisition rate depends on the trigger arduino

save_to_video = True  # ! decide if you want to save the videos or not
```

The first 3 are the only one that need to be changed for each experiments, the others can generally be left as they are.

- **Experiment_folder**: the folder where the files for the experiment (videos, csv files, figures...) are saved.
- **Experiment_name**: this is used to uniquely identify each file associated with one experiment. If there are already files in experiment_folder that start with experiment_name, this will give an error to prevent overwriting the data.[this option can be diasbled by setting overwrite_files as True, but I advice against that].
- experiment_duration: pre-determined duration of the experiment in seconds [see more details below]. To disable tis option, set is as None.

After the forceplate_config file is correctly configured, **starting an experiment is simply a matter of running main.py**. In **main.py** you don't need to actually change any settings as all the options are specified in forceplate_config.

To **stop an experiment** there are two options.

1. in forceplate_config, the user can set "experiment_duration" to the number of seconds she wants the experiment to last. Once that ammount of time is elapsed the experiment is terminated. To prevent this from happening, you can set experiment_duration = None and that will disable this option, allowing for experiments of arbitrary duration.
2. Alternatively at any time you can manually stop the experiment by entering Ctrl-C in the python editor in Visual Studio Code.

Upon termination of an experiment (unless something goes wrong), two figures will be generated: one showing the delay between frames during the acquisition, the second showing the readouts from the force sensors during the

experiment.

During an experiment

A brief overview of what is happening during an experiment, i.e. how the data are actually acquired.

As part of data acquisition a number of files are saved: one video file for each camera (ending in "cam0.mp4"..), one .csv file with the sensors readouts. One .txt file with information about the videos being generated and potentially a number of figures saved as .png upon experiment termination.

Part 1 - Setup

1. When the experiment is started, a folder with path experiment_folder is created if it doesn't exist
2. The code checks if files with experiment_name exist already in experiment_folder, if they do it gives an error to prevent overwriting data
3. A .csv file is created in which the sensor data will be stored
4. Connection to the sensors arduino is initialised
5. Cameras and ffmpeg video writers are initialised
6. Data acquisition starts

Part 2 - Acquisition

Data acquisition happens in a loop, at each loop a frame from each camera is acquired and the state of the sensors is read and saved to the .csv file. These are the steps that happen at each loop:

1. The trigger arduino sends a trigger to both cameras
2. Both cameras acquire a frame simultaneously and save it on their local memory (buffer)
3. The python code gets the frame from each camera and sends it to the corresponding ffmpeg video writer who appends the frame to the video being generated.
4. The python code, through pyFirmata, reads the state of the analog inputs [force sensors states] and saves them in a new line of the .csv file together with the frame number and a timestamp.
5. If the option "live_display" is set as true, the frames are displayed during acquisition. This slows down data acquisition considerably so I suggest avoiding that.

Analysis

Once data acquisition is terminated, the user can further analyse the data using the scripts in the Analysis folder. Currently three types of analysis can be performed: 1) Manual video inspection - looking at a video frame by frame, 2) Plotting stuff (e.g. sensors readouts...) and 3) Video analysis → the creation of summary videos that incorporate the video frames and a number of plots.

Analysis Config options

These can also be found in forceplate_config, they need to be set correctly before running any of the analysis scripts, with the exception of manual_video_inspection.

```
analysis_config = {
        "data_folder": "E:\\Egzona\\180719", # where the data to analyse are stored
        "experiment_name": "180719_M1R",
```

```
        "plot_colors": { "fr":magenta,
                         "fl":blue,
                         "hr":red,
                         "hl":green},

        # * for composite video
        # ? run video_analysis.py
        "start_clip_time_s": 368, # ? Create clips start at this point, in SECONDS
        "start_clip_time_frame": None, # ? Create clips start at this point, in FRAMES
        "clip_n_frames": 368, # duration of the clip in frames
        "clip_name":"",

        "outputdict":{ # for ffmpeg
                    # '-vcodec': 'mpeg4',  #  high fps low res
                    "-vcodec": "libx264",   #   low fps high res
                    '-crf': '0',
                    '-preset': 'slow',  # TODO check this
                    '-pix_fmt': 'yuvj444p',
                    "-framerate": "10", #   output video framerate
                    # TODO this doesnt work FPS
                },
    }
```

- Data folder and experiment name work in the same way as for data acquisition above. In this case they are used to find all the files (video, .csv ...) that belong to the experiment being analysed.
- Plot colours shows the colour associated with each sensor. Colors are specified in the file "*Utils\constants.py*" as RGB values.
- start_clip_time_s and start_clip_time_frame are used to indicate the start time of the composite video compared to the whole recording. This can be expressed in seconds from the start of the video, or in frames. To use frames, you'll have to set start_clip_time_s as None.
- clip_n_frames: is the number of frames you want use to create the composite video. The composite video will start at the frame number specified above, and go over all the frames of the experiment's videos and and .csv files, it will stop when it iterated over the number of frames specified here.
- clip_name: is the name to be used for saving the composite file. The composite file will be saved as "experimentname_clipname.mp4". If a file with this name exists already in the data_folder, this will throw an error to prevent overwriting data.

Analysing data

Manual video inspection

To look at a video frame by frame you can run **manual_video_inspection.py** in the analysis subfolder. There you need to specify the path to the video you want to look at. For this script you don't need to set up anything in forceplate_config.

When the script is run it will show you the first frame of the video selected. You can move backward and forward along the video with the letters "a" and "d", and you can skip to an arbitrary frame by pressing "s" and then entering the desired frame.

Plotting

To create summary plots of the data acquired during an experiment, after correctly setting up the options showed above, run **analysis.py** in the analysis subfolder of APA_setup.

Currently the only plot being generated simply shows the readouts from the force sensors during the experiment, but we can add more plots if desired.

Creating composite video

This allows for the creation of a video which shows the frames from the camera together with a number of plots illustrating the state of the sensors and in the future the pose reconstructed with DLC.

To do this you'll have to set up all the options in forceplate_config shown above, then you can run **video_analysis.py** inside the analysis subfolder. *

* in the future this process might be automated to generate composite videos from notes saved on an excel file during manual video inspection.