# Design brief:

To generate supervised machine learning models to detect specific animal behaviours, we require a training dataset of labelled example behaviours. These are typically generated by filming an animal wearing a device and then time matching instances from the accelerometer ‘trace’ with the videos. There is no single way this is achieved, and each project brings with it specific problems that often require custom solutions. In this case, we face two main problems:

1. Camera timestamp and accelerometer timestamps are mismatched
2. Accelerometer timestamp ‘drifting’ over time (progressively more incorrect)

To overcome these, we need a system that allows us to load in a trace and a video. We align the first marker (normally a ‘clap’) which sets the absolute difference between trace and video times. We then set this as time 0 for both media formats. Then when we load in subsequent videos, their position is calculated based on the relative difference from the original video. There needs to be a fair bit of flexibility though as the drift over time means each alignment jump will be progressively but unpredictably worse.

# Earlier work:

The sync\_station\_v2 that I am using as my starting base was designed by Chris Clemente in 2024 (included in the folder). It is an updated iteration of a program originally conceived and designed 2017-2021, which can be found at the old git repo: <https://github.com/cclemente/Animal_accelerometry>. The limitation with this method is that, since the introduction of the ‘zoom’ functionality, the time-jumps seems to have been decoupled.

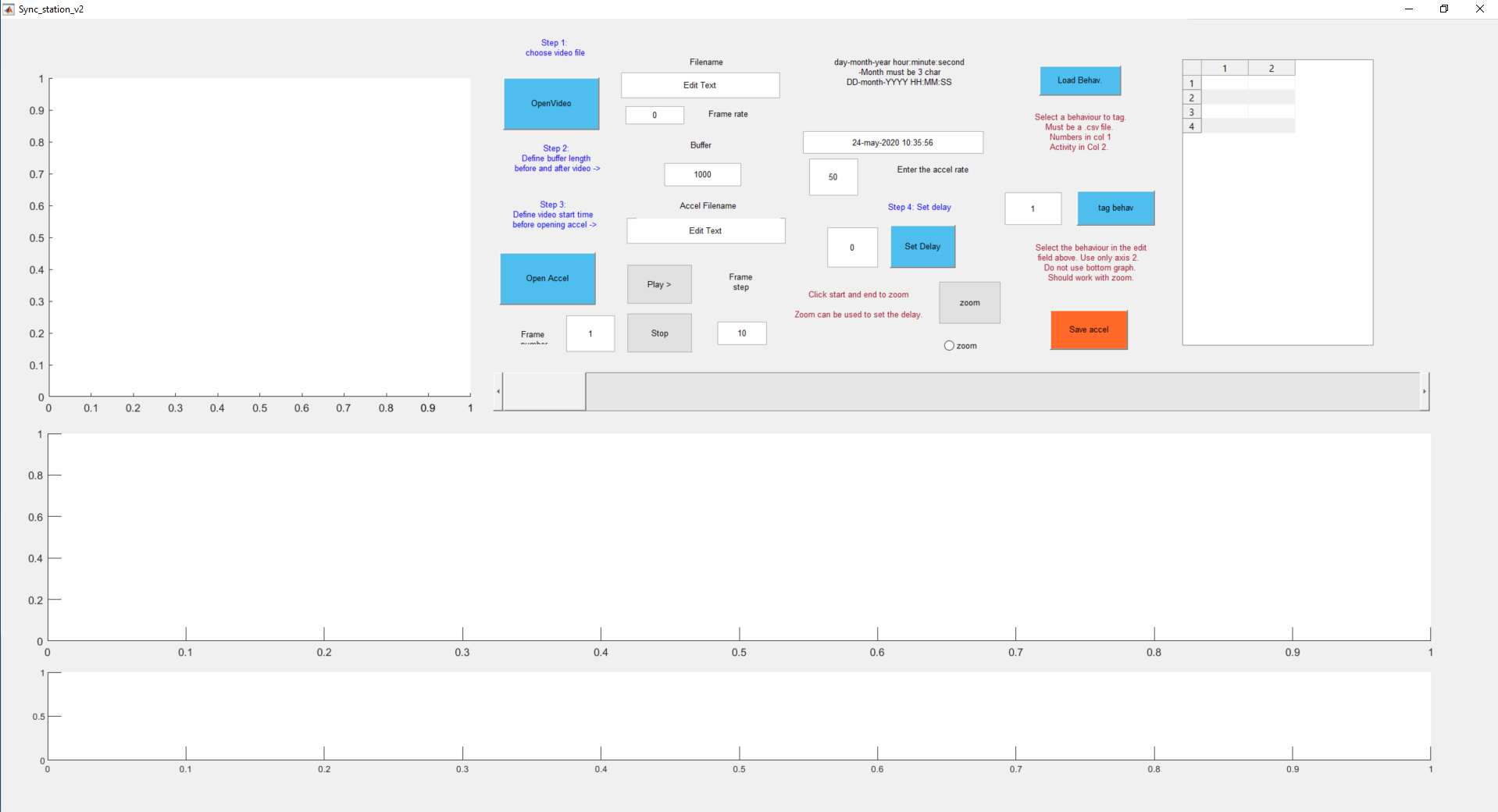
# Worklog:

22/10

* I am going through the existing code to familiarise myself, update the formatting to structured functions, remove outdated code/comments, and rename variables to more obvious names.

|  |  |  |  |
| --- | --- | --- | --- |
| *Name* | *Type* | *Functionality* | *Original name* |
| *GUIDE objects* | | | |
| video\_display | Display |  | axes1 |
| large\_accelerometer\_display | Display |  | axes2 |
| small\_accelerometer\_display | Display |  | axes3 |
| load\_accel\_button | Button | Browse for accel file | pushbutton2\_accel |
| load\_video\_button | Button | Browse for video file | pushbutton1\_video |
| video\_name\_text | Text display | Displays the name of the video file | edit1 |
| vid\_frame\_rate\_text | Text display / edit | Video frame rate | edit4\_getframe |
| set\_buffer | Text entry | Additional accelerometer frames to display | edit3\_buffer |
| accel\_name\_text | Text display | Displays the name of the accelerometer file | edit2 |
| play\_button | Button | Plays video | pushbutton3\_forward |
| stop\_button | Button | Stop video | pushbutton4\_stop |
| current\_frame | Text display | Displays current frame | edit\_framenum |
| set\_frame\_step\_value | Text entry | Enter the number of frames skipped in each play | edit5\_frame\_step |
| set\_accel\_start\_datetime | Text entry | Set the datetime of the file: day-month-year hour:minute:second  -Month must be 3 char  DD-month-YYYY HH:MM:SS | edit\_acceldate |
| set\_accel\_frame\_rate | Text entry | Set the Hz of the accelerometer | edit\_accelrate |
| delay\_text | Text display | Reads the delay between video and accel | edit\_delay |
| delay\_calculation\_button | Button | Finds current difference and sets to delay\_text | pushbutton\_setdelay |
| zoom\_trigger\_button | Button | Triggers 2 clicks that will bound the large\_accelerometer\_display | pushbutton\_zoom |
| zoom\_toggle | Radiobutton | Toggle whether zooming has already been done | radiobutton1\_zoom |
| load\_behaviours\_button | Button | Load the csv file | pushbutton7 |
| behaviours\_table | Table | Display the behaviours csv | uitable1 |
| set\_behaviour\_number | Text entry | Set the number corresponding to selected behaviour | edit\_behnum |
| annotation\_trigger\_button | Button | Triggers 2 clicks that will annotate the trace | pushbutton\_tagbeh |
| save\_accel\_button | Button | Saves the annotated accel trace (as the name of the video) | pushbutton\_save |
| video\_slider | Slider | Slide through the video – also moves the vline in large\_accelerometer\_display | slider1 |
| *Functions* | | | |
| display\_small\_accel\_fun |  |  | mydisplay3 |
| display\_large\_accel\_fun |  |  | mydisplay2 |
| display\_video\_fun |  |  | mydisplay |
| initialise\_gui\_fun |  |  | Sync\_station\_v2\_OpeningFcn |
| print\_to\_terminal\_fun |  |  | Sync\_station\_v2\_OutputFcn |

* While the ‘GUIDE’ figure designer is useful for aesthetics of the app, it is soon to be removed from MatLab. Therefore, we need to update it either to ‘App Designer’, or we can replace it with a static programmatically coded version – which seems preferable.
  + Initially tried stripping out all the information from the existing figure using fig2m, code which I found here: <https://au.mathworks.com/matlabcentral/fileexchange/14340-convert-fig-to-matlab-code> (including debugging information in reviews) but was unable to make it functional due to lack of UIcontextMenu in the figure.
* Approaching it manually, I grouped existing bits into functional units, each stored in their own script. These are indicated in the colours below. These are then drawn together into the main script with an initialisation command. Before I make any changes to the math and functionality, I want simply to recreate the original version.



23/10

* Today I will group the functional groups into scripts, then make them functions that can be called one by one (after checking that a basic version works)

|  |  |  |
| --- | --- | --- |
| *Colour* | *Functionality* | *Script name* |
|  | Select video and set frame rate | VideoLoad |
|  | Display the video, including play, pause and frame-step | VideoDisplay |
|  | Select accel, set frame rate, and enter the start date/time of the accel | AccelLoad |
|  | Display the large and small accelerometer sections | AccelDisplay |
|  | Zoom in and out of the large accelerometer display | ControlAccelZoom |
|  | Align the video with the trace and set delay | TimeAlignment |
|  | Select the csv with the behaviours loaded | BehavioursLoad |
|  | Enter the behaviour number as well as trigger annotations | BehavioursAnnotate |
|  | Save the annotated accelerometer csv | AccelSave |

* I had a chat with Luke (Jessup, somewhat postdoc in out lab) who has used Matlab extensively and he suggested that rather than going back to basics with programmatic design, I could use App Designer. Chris then said I can do a lower-level version of it, we can then send it to a real software engineer to convert to a C++ exe. Therefore, App Designer it is.
* <https://au.mathworks.com/help/matlab/app-designer.html>
* Going to work through adding one component at a time.
  + Seems to work like – you produce an object and then this adds to the script. You can then do several things to the code. A Callback is a function that actions when it is triggered – generally by the user doing something.
  + Video can only be shown frame by frame as a series of images
  + No need to use handles in app designer, we can access objects directly. Instead of handles.myButton we just do app.myButton
* When I redesign it, I would like to manually set the start time of both the video and the accelerometer and have options to auto-align and manual align

