OS Tracking Manual

Written by Ping Hill, July 2023
Email: soongpinghill87@gmail.com

Compression:
https://colab.research.google.com/drive/1Z-7BaEj3ySd4DnGWIN6DbJANOTfOd3zH?usp=sharing

Synchronization + Extraction + Analysis
https://colab.research.google.com/drive/1nVy4tAC1-F6IGkVaOgOlqwWQWIRoGiNk?usp=sharing

Google drive folder example of point grey, videos, signals :
https://drive.google.com/drive/folders/1gAA5G5rxCyFRozf42xF3Vtq2kR52b1Ds?usp=sharing

Google drive folder example of Deeplab cut, including training data-sets, dlc models, etc
https://drive.google.com/drive/fold1ers/1uO14Jg19VbM8GbI0heytpwT933XCS9n4?usp=drive_link

There are 4 Important steps needed for successful OS Tracking

- Compression
- Synchronization
- Extraction
- Analysis

Compression

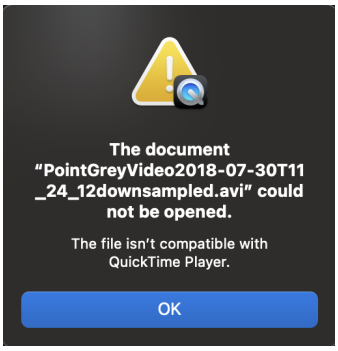
Compression is necessary as a single video can be up to 7+ hours and more than 10 GB in size. For this task, I implemented using FFMPEG to compress my videos. You can download it here, and get the version suitable for your computer. This is the tutorial that provided me with steps on how to download it: video.

This step only utilizes 3 functions, “find_downsampled’, “find_finished_total”, and “downsample_stuff”. The code utilizes a for loop to look for files in your folder that end in .avi (indicating it is a video), and uses another loop to compress.

```
def find_downsampled():
  all = []
  for root, dirs, files in os.walk("/content/drive/MyDrive/Colab Notebooks/ RGS14"):
      for file in files:
          if file.endswith(".avi"):
            if 'downsampled' in file:
              all.append(os.path.join(root, file))
  return all
```

This is an example of the first function in “Compression”. The only thing that should be altered is the file location. I had my folder on google drive in a folder named RGS14, so that is what I designated. After running this code and designating the correct file containing your videos for all 3 functions, you should get a video similar to this:

PointGreyVideo2018-07-30T11_24_12downsampled.avi



This is your downsampled video, the resolution and dimension have been lowered, and this is the video we will use for tracking. Note, if using Google Colabs, this video will be found and available for download in the same folder as what was designated in the code.
If an error like this occurs, simply download/ utilize VLC, and drag the video in the workspace to access it.

Synchronization.

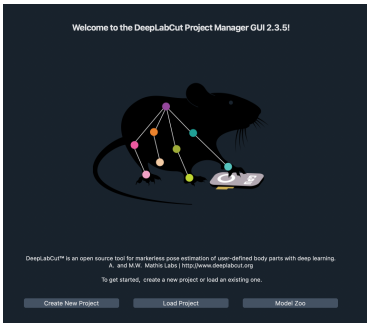
In the process of synchronization, we utilize deeplabcut to find the appropriate trials in the video. When analyzing and tracking videos with deeplabcut, it provides a likelihood that the desired coordinate (head) is in the frame. By converting the output of the tracker into a CSV file (including coordinates and likelihoods), we can filter out frames that do not have a high likelihood of the rat, thus making sure we only include frames with the rat. This will be explained more in-depth later.

The first step for Synchronization is downloading deeplabcut. There are many ways to do this, but the most straightforward is following the steps here or following this tutorial. It will ensure you have Anaconda, and if there are any issues, make sure systems like Pytorch, TensorFlow, python, etc are updated/ in a version suitable with deeplabcut. This is also a nice tutorial to learn the basic commands of DLC:

Once deeplabcut is downloaded, in your terminal you can type:

"**Pythonw -m deeplabcut**"
and the application will open.

Setting up Training



After opening, you will be greeted with this screen. They will ask you who is the experimenter (put down your name), the project name, and the date created. It will then ask to choose a folder, choose one that includes the downsampled video or videos that were previously compressed.
As soon as a new project is created, a folder will be created with "project+name+date"
Inside this folder, there will be a "config_yaml file", "dlc_models", "training_datasets", "labeled_data" and "video's folder". Once you have created the project, there is no need to stay on this application as you don't need to extract/ label frames, as we will import pre-labeled frames for training.

To import the files for training, navigate to this GitHub link, and download the zip file. From here you can see the files in the GitHub link also include “config_yaml file”, “dlc_models”, “training_datasets”, “labeled_data” and “video.
You want to drag the files “dlc_models”, “training_datasets”, and “labeled_data” from this downloaded github link to your own created folder, replacing the empty folders of the same name. This will allow you to train your network on a vast amount of pictures and diverse frames. You dont need to replace your config file, but edit it so that it looks like this:

```yaml
scorer: ping
date: Jun15
multianimalproject: false
identity:

    # Project path (change when moving around)
project_path: /Users/iphone10/Desktop/48good-ping-2023-06-15

    # Annotation data set configuration (and individual video cropping parameters)
video_sets:
  /content/drive/My Drive/48-ping-2023-06-15/videos/PointGreyVideo2018-07-30T11_24_12downsampled.avi:
    crop: [0, 325, 0, 256]
bodyparts:
- Head
- Sideleft
- Sideright
- Spine1
- Tailbase
- Tailmiddle

# Fraction of video to start/stop when extracting frames for labeling/refinement

    # Fraction of video to start/stop when extracting frames for labeling/refinement
start: 0
stop: 1
numframes2pick: 20

    # Plotting configuration
skeleton:
- [Head, Sideleft]
- [Head, Sideright]
- [Sideleft, Spine1]
- [Sideright, Spine1]
- [Spine1, Tailbase]
- [Tailbase, Tailmiddle]

skeleton_color: white
pcutoff: 0.6
dotsize: 3
alphavalue: 0.7
colormap: autumn

    # Training,Evaluation and Analysis configuration
TrainingFraction: [0.95]
iteration: 0
default_net_type: resnet_50
default_augmenter: default
snapshotindex: -1
batch_size: 8

    # Cropping Parameters (for analysis and outlier frame detection)
cropping: false
    #if cropping is true for analysis, then set the values here:
x1: 0
x2: 640
y1: 277
y2: 624

    # Refinement configuration (parameters from annotation dataset configuration also relevant in this stage)
corner2move2: [50, 50]
move2corner: true
```

With the video_set to be where the downsampled video is stored (should be in the videos folder of the deeplabcut created folder). My location has it set to google drive as this is where I stored it. Make sure to change the body parts and skeleton.

From here, you should have your created deeplabcut folder with your original but edited config_file, and video file, but should have replaced the dlc_models", "training_datasets", and "labeled_data" folders with those from GitHub.

In the dlc_models folder, navigate to iteration 0 (you will end up doing this step for all 4 iterations), continue to project+date+shuffle1, train, then pose_cfg.yaml.

- In pose_cfg.yaml, navigate to the line that states "project_path", and paste the path of your project. In my case, I had copied the whole folder into google drive, so it looked like this:

project_path: /content/drive/My Drive/48good-ping-2023-06-15.

For setting the init_weights, you need to make sure you have the deeplabcut github folders downloaded somewhere on your computer. You can find it here. You then follow

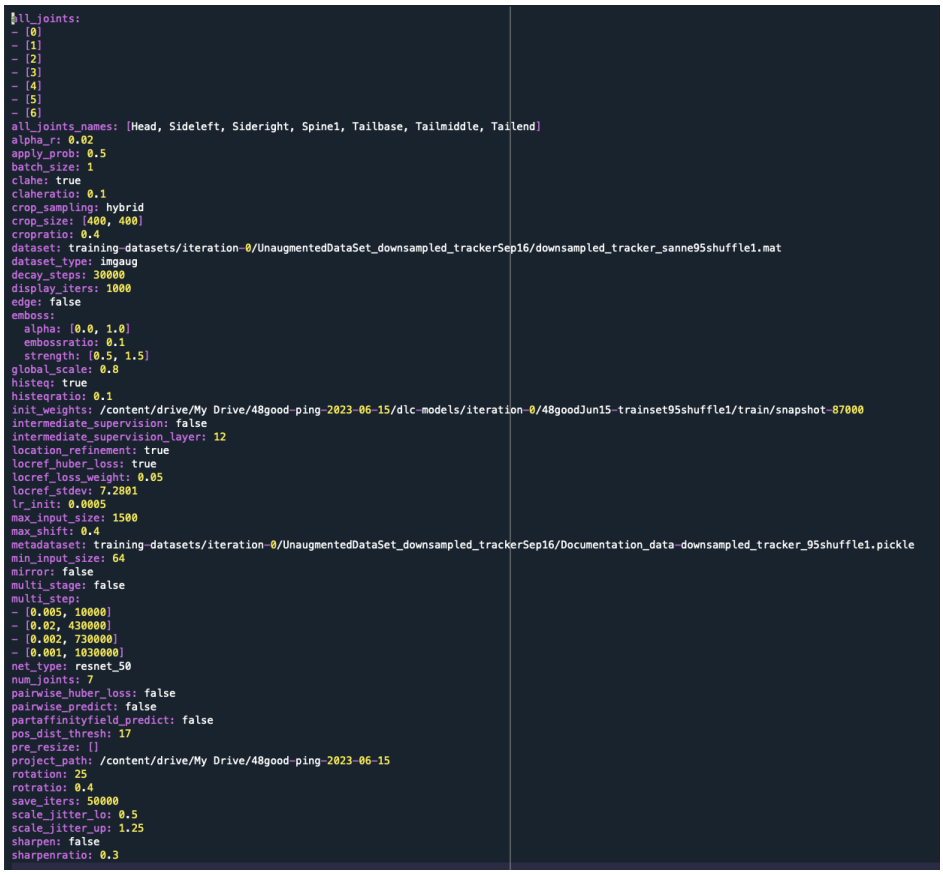
DeepLabCut/deeplabcut/pose_estimation_tensorflow/models/pre-trained/pretrained_models_url.yaml. Pathway.

- Find the line resnet_50, and click on the link to download the pre-trained model. I then dragged this resnet_v1_50.ckpt into my pre-trained folder that I had downloaded

After downloading this resnet file, make sure to set the init_weights are set to the location where you have the folder/ resnet file location. In my case, it looked like this:

Init_weights:
/content/drive/MyDrive/deeplabcut/pose_estimation_tensorflow/models/pre-trained/resnet_v1_50.ckpt

This is what my Pose_cfg.yaml file looks like. If any of these pathways are incorrect, you will run into many issues. One of which will be files not found. Here my init_weights is from a previous trained point.



```
all_joints:
- [0]
- [1]
- [2]
- [3]
- [4]
- [5]
- [6]
all_joints_names: [Head, Sideleft, Sideright, Spine1, Tailbase, Tailmiddle, Tailend]
alpha_r: 0.02
apply_prob: 0.5
batch_size: 1
clahe: true
claheratio: 0.1
crop_sampling: hybrid
crop_size: [400, 400]
cropratio: 0.4
dataset: training-datasets/iteration-0/UnaugmentedDataSet_downsampled_trackerSep16/downsampled_tracker_sanne95shuffle1.mat
dataset_type: imgaug
decay_steps: 30000
display_iters: 1000
edge: false
emboss:
  alpha: [0.0, 1.0]
  embossratio: 0.1
  strength: [0.5, 1.5]
global_scale: 0.8
histeq: true
histeqratio: 0.1
init_weights: /content/drive/My Drive/48good-ping-2023-06-15/dlc-models/iteration-0/48goodJun15-trainset95shuffle1/train/snapshot-87000
intermediate_supervision: false
intermediate_supervision_layer: 12
location_refinement: true
locref_huber_loss: true
locref_loss_weight: 0.05
locref_stdev: 7.2801
lr_init: 0.0005
max_input_size: 1500
max_shift: 0.4
metadataset: training-datasets/iteration-0/UnaugmentedDataSet_downsampled_trackerSep16/Documentation_data-downsampled_tracker_95shuffle1.pickle
min_input_size: 64
mirror: false
multi_stage: false
multi_step:
- [0.005, 10000]
- [0.02, 430000]
- [0.002, 730000]
- [0.001, 1030000]
net_type: resnet_50
num_joints: 7
pairwise_huber_loss: false
pairwise_predict: false
partaffinityfield_predict: false
pos_dist_thresh: 17
pre_resize: []
project_path: /content/drive/My Drive/48good-ping-2023-06-15
rotation: 25
rotratio: 0.4
save_iters: 50000
scale_jitter_lo: 0.5
scale_jitter_up: 1.25
sharpen: false
sharpenratio: 0.3
```

As of right now, the deeplabcut folder should contain your config_yaml file, your video folder, the 3 “dlc_models”, “training_datasets”, and “labeled_data” folders, and inside dlc_models → iteration0-3 —> projectname+trainsetshuffle → train → pose_yaml, you have edited the correct project path and init_weights.

Synchronization:

Point Grey files

- You should have been given access to the point grey files that correspond to the video
- These include
    - Point Grey Frame number
    - Point Grey Timestamps
    - Bonsai Timestamps
    - Point Grey LED status
    - Other files

ADC files

- You should also be given access to ADC following this format, we only take into account ADC1
    - 100_ADC1_0.continuous
        - (One little-endian int64 timestamp (actually a sample number; this can be converted to seconds using the sampleRate variable in the header)
    - 100_ADC2_0.continuous
        - One little-endian uint16 number (N) indicating the samples per record (always 1024, at least for now)
        - Not used

Extraction_t function: takes in file_path, date_serial, ADC='1', ADC_loc=''.

The file_path is the file with all the ADC and point grey files, and date_serial is the date of the trial. ADC will always be set to 1, ADC_loc is used to specify which trial it is. If it is trial 2, you state ADC_loc = 2, (assuming it is in a folder labeled "2" and this line will direct the location. Such as:

ADC_File = "ADC_loc+'100_ADC'+'1'+'_0.continuous'"

If there are no different folders for trials, it is okay to leave ADC_loc empty initially.

The first few lines are loading in and initializing data. Load in all the point grey and ADC files, but we are only going to pay attention to DF_P_LED, and DF_LED_ADC.

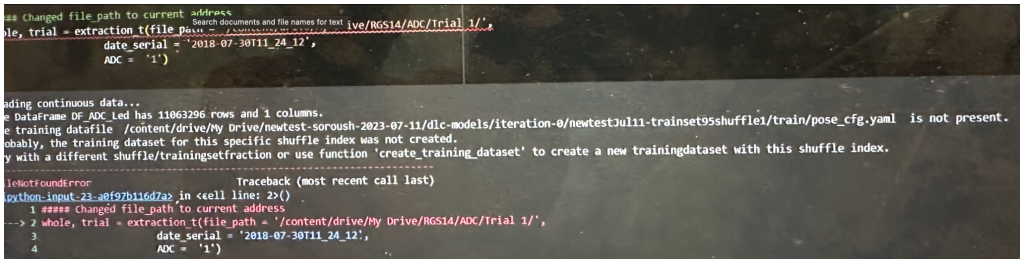
```
ProjectFolderName = '48good-ping-2023-06-15'
VideoType = 'avi'
path_config_file = '/content/drive/My Drive/'+ProjectFolderName+'/config.yaml'
t_video = file_path+VIDEO_file
t_video = t_video.replace('.avi', '-TRIM.avi')
#deeplabcut.create_training_dataset(path_config_file, net_type='resnet_50', augmenter_type='imgaug')
#deeplabcut.train_network(path_config_file, shuffle=1, displayiters=500, saveiters=500, maxiters=100,000, keepdeconvweights= True)
deeplabcut.analyze_videos(path_config_file, t_video, save_as_csv=True)
```

In the next section of code, initialize the folder name that you created in deeplabcut, specify the video type, and update the path to the config file. I did this through google drive. This is when you can start training your dataset, and make sure you have imported/ downloaded deeplabcut as a module.

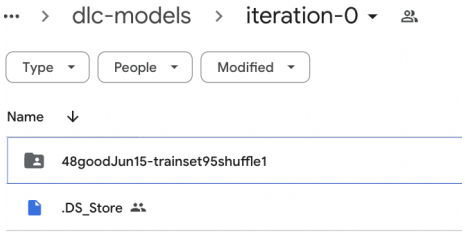
Use create_training_dataset, and specify the net_type you are using, in our case, it's resnet_50.
Use train_network to start training your network, pass in the path to the config file, displayiters is how often you want it to show the progress, saveiters indicate how many iterations to pass every time it's saved (in your DLC-models folder), and maxiters is when you want it to end. I recommend training for 50,000-100,000 iterations.

If you must pause training, keepdeconvweights lets you start from a pre-saved spot. make sure you have saved the progress, and know where the training iterations are being saved. When starting up your training again, make sure you update your init_weights (in the training dlc_models → pose_cfg.yaml file) to where the previously saved iterations were.

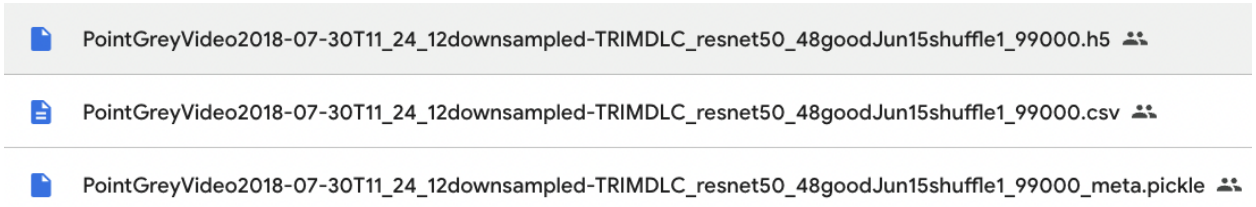
In my case, they were stored in my deeplabcut created folder → dlc-models

- Init_weights
    - /content/drive/My Drive/48good-ping-2023-06-15/dlc-models/iteration-0/48goodJun15-trainset95shuffle1/train/snapshot-87000
- If you come across an issue like this, where they are unable to find the pose_cfg file:
- 

    - Go into your dlc-models/ iteration 0 / Change the name of the folder to match what the code says it is. It should be: name+date - trainset95shuffle1. This has to be manually done due to the use of another dataset used for training.



After training, many files will be created in your folder holding point grey and ADC files, including a .h5, .csv, and .pickle file. To continue onto the next step, you must have the .h5 file. They will look like this:



99000 is the number of iterations I trained for.

The next section of code is utilizing the analysis of the tracking (finding the likelihood that the rat is in the frame) and cuts it based on parameters (if the likelihood is more than 0.99 for 1000 continuous frames, it is included. We do this by utilizing a dictionary and looping through it. This is still all in the same extraction_t function, however, when testing you can do it separately.

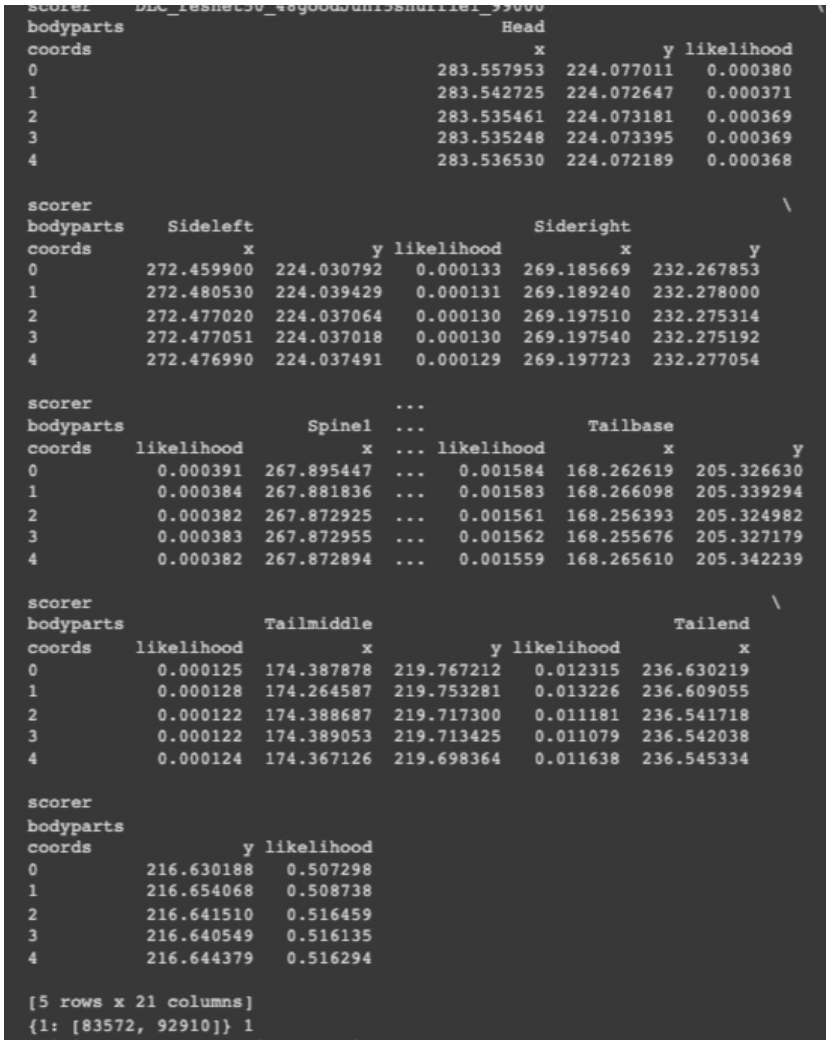
You start this section of code initializing the DLC scorer. It is essentially a data frame with the coordinates of body parts and likelihood It will have the format of DLC_resnet50+project+date+shuffle+iteration. In my case this is what it looked like: DLCscorer='DLC_resnet50_48goodJun15shuffle1_99000'

We then load the output of the tracking.

This line:

df = Dataframe['DLC_resnet50_48goodJun15shuffle1_99000']['Head']['likelihood'].to_frame()

Creates a data frame with the x coordinates and the likelihood it is in the frame. Make sure to change the CSV file to the name of yours. The following lines of code execute what was explained earlier, finding the right frames and cutting the video based on the timestamps.
The output of this segment of code will be a mp4 video roughly 5-6 minutes long, however not yet tracked. There are also print statements throughout to check on the progress of this cutting and sorting. This output can be seen here:

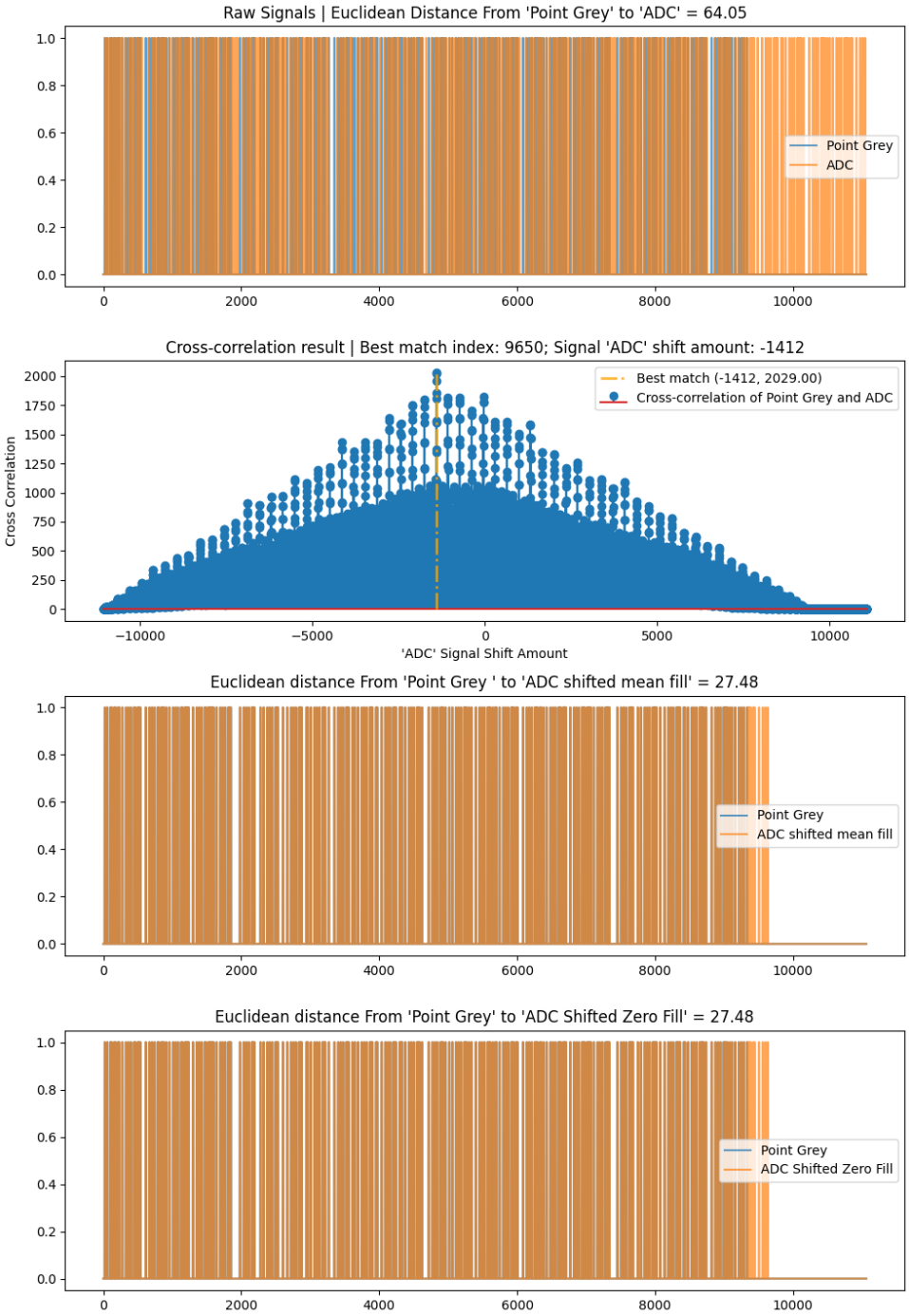


```
bodyparts                                  Head
coords                                        x           y likelihood
0                                     283.557953  224.077011   0.000380
1                                     283.542725  224.072647   0.000371
2                                     283.535461  224.073181   0.000369
3                                     283.535248  224.073395   0.000369
4                                     283.536530  224.072189   0.000368

scorer                                                                \
bodyparts    Sideleft                              Sideright
coords              x           y likelihood             x           y
0          272.459900  224.030792   0.000133    269.185669  232.267853
1          272.480530  224.039429   0.000131    269.189240  232.278000
2          272.477020  224.037064   0.000130    269.197510  232.275314
3          272.477051  224.037018   0.000130    269.197540  232.275192
4          272.476990  224.037491   0.000129    269.197723  232.277054

scorer                          ...
bodyparts                Spine1 ...              Tailbase
coords    likelihood          x ... likelihood          x           y
0           0.000391  267.895447 ...   0.001584  168.262619  205.326630
1           0.000384  267.881836 ...   0.001583  168.266098  205.339294
2           0.000382  267.872925 ...   0.001561  168.256393  205.324982
3           0.000383  267.872955 ...   0.001562  168.255676  205.327179
4           0.000382  267.872894 ...   0.001559  168.265610  205.342239

scorer                                                                \
bodyparts             Tailmiddle                                Tailend
coords    likelihood          x           y likelihood          x
0           0.000125  174.387878  219.767212   0.012315  236.630219
1           0.000128  174.264587  219.753281   0.013226  236.609055
2           0.000122  174.388687  219.717300   0.011181  236.541718
3           0.000122  174.389053  219.713425   0.011079  236.542038
4           0.000124  174.367126  219.698364   0.011638  236.545334

scorer
bodyparts
coords              y likelihood
0          216.630188   0.507298
1          216.654068   0.508738
2          216.641510   0.516459
3          216.640549   0.516135
4          216.644379   0.516294

[5 rows x 21 columns]
{1: [83572, 92910]} 1
```

At the bottom, you can see the dictionary created. The first trial was found at frames 83572 and ended at 92910. Using a frame-to-second calculator, with 30fps, this translated to 46min 25sec - 51min 37 sec, exactly when the first trial started. For this code, it only shows the first trial it finds. In future student projects, the code must be configured to display and cut all trials in the video, most of the time it is 5 trials a day.

The following lines of code will synchronize the two signals together (DF_P_LED and DF_LED_ADC)

- DF_P_LED will contain all the LED signals for the whole video (can be data of up to 7 hours), so it is necessary to cut it as we did earlier
- DF_LED_ADC is its own file, recorded separately and 1 trial at a time, there is no need to edit the length of this, we just must synchronize it.
- Utilizing compare_and_plot_signals_of_alignment, we a
    - This function first checks if the signals are the same length, if not, it pads the shorter one with 0's to make them the same
    - It then cross-correlates the two signals, finds the best match, and shifts the signal (by filling with either 0's of the mean) to match the indices.
    - We are given 4 graphs, 1 with the raw un-synchronized signal, 1 showing the cross-correlation, and 2 showing the synchronized signals.
- The output should look like this:



Plot 1: the raw signals of a and b, unaligned

- the higher the Euclidian distance, the worse synchronized it is

Plot 2: Cross-correlation.

- X-axis: ADC shift amount relative to PG. represents all possible shift positions for ADC to shift.

- Y axis: cross-correlation values between PG and ADC, indicates the degree of similarity between the two. The higher the value the better the correlation

- Best match index: index in cross-correlation result with highest correlation value, in this case, the index is 9650, while the correlation value itself is 2029.
- signal ADC shift amount is how much to shift the signal to match the Point Grey signal. (In this case it's-1412)

plot 3:

- alignment between a and b when shift is achieved by filling the shift with the average of b.

plot 4:

- alignment between a and b when the shift is achieved by filling the shift with 0.

This should have successfully shifted and synchronized the two signals together. This is an example of how you could call such a function:

```
whole, trial = extraction_t(file_path = '/content/drive/My Drive/Colab Notebooks/RGS14/',
                date_serial = '2018-07-30T11_24_12',
                ADC =  '1')
```

After this step, we can finally create a labeled video: It is important to wait until the very end to create the labeled video, as there are many issues when attempting to create one directly after training.
This is the function:

- def main_deeplabcut(directory, config, video type, save_csv, create_labeled, create_plots, analyze_skeleton):

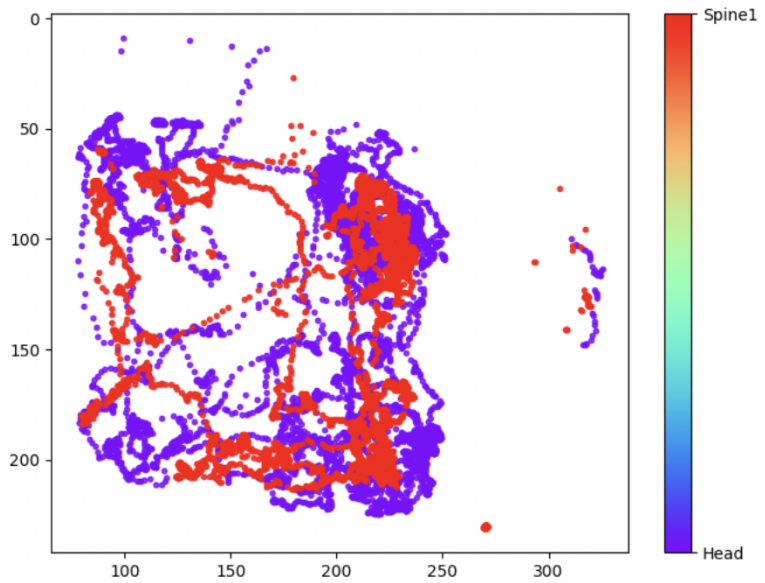
An example of how to call it:

- main_deeplabcut('/content/drive/My Drive/Colab Notebooks/RGS14/trimmed', path_config_file, VideoType, True, True, True, True)

Analysis of tracking:

After creating the labeled video, we can perform analysis on the rat's movements, including the speed, where the rat goes, the average distance between body parts, etc, more analysis and function can be written in the future, such as the angle between head and spine, and of course, actually correlating with the spike data to see real-time what is going on.

An example of tracking the rat in the area:



An example of the average distance between body parts used mainly to test the accuracy of the tracker:

	x	y	x_sideleft	y_sideleft	distance
0	269.185669	232.267853	168.262619	205.326630	104.457128
1	269.189240	232.278000	168.266098	205.339294	104.456567
2	269.197510	232.275314	168.256393	205.324982	104.476932
3	269.197540	232.275192	168.255676	205.327179	104.477056
4	269.197723	232.277054	168.265610	205.342239	104.464232

53.982810292037534

Next steps:

The next step would be to make sure all trials are found and cut correctly. As stated earlier, the script only finds the first trial and cuts it, but for this to be successful, it needs to find all. There have also been some issues with the videos being cut too early in the beginning, and we are shown up to 20-30 seconds of a blank arena. This could be due to the rat being introduced and pulled back really fast, but still doesn't account for the 3-5 seconds before the rat is introduced at all.

Some further changes that will be implemented are getting rid of tracking the last tail. Tracking this tail causes the tracker to mistake the wire to be part of the mouse, and compromises the tracking data

and accuracy. We further will change the settings of the tracker, make the colors more visible, and make the points bigger.