Getting Started

Pierre Quentin Ngandjui Tiako

September 8, 2024

Abstract

This document provides a comprehensive guide to getting started with the Thesis code: Machine Learning based High Level Control. This document is designed to explained the data used, what are the different part of the code, from data filtering to Machine Learning model training.

1 Introduction

Welcome to the Machine Learning based High Level Control getting started guide. This document is designed to help new users quickly and efficiently begin using the code and data used for this Thesis.

2 Prerequisites

Before you begin, ensure you have the following prerequisites:

- A computer with internet access
- An operating system (Windows, Linux)
- Python install via your prefererred environnment manager (Conda, ...), the Python version used in this project is the 3.9
- Package installer for python (pip will be the example in this document)
- Install the different packages needed with this command line in the terminal

```
pip install -r requirements.txt
```

3 Datasets

Here are the different original databases used in this project which are in the Experimental_Data folder in the Original_Databases folder:

Table 1: Databases parameters: One experiment per Walking speed

Parameters	**[2] of K. Embry et al:**	**[1] of Fukuchi et al:**	**[3] of Moreira et al:**
Folder Name	InclineExperiment	WBDSData	MAT files
Number of subjects	10	42	16
Height range (in m)	1.57 to 1.86	1.71 ± 0.105	1.51 to 1.83
Mass range (in kg)	47.8 to 75.0	68.4 ± 12.2	52.0 to 83.7
Age range (in years)	19 to 27	45 ± 10.4	20 to 28
Walking speed (in m/s)	0.8, 1.0, 1.2	0.23 to 2.23	0.28 to 1.1
Gait cycle per experiment	42	1	10

The data organization for each of these databases is different. The functions for extracting this data will be described below. The datasetV2.h5 file is the one obtained after extracting all the data from the original databases and saving them according to the 1 structure. The processed_data file is the one where only the trajectories corresponding to a ground inclination of zero and also the trajectories where it has been calculated that the key points defined in this project can be found.

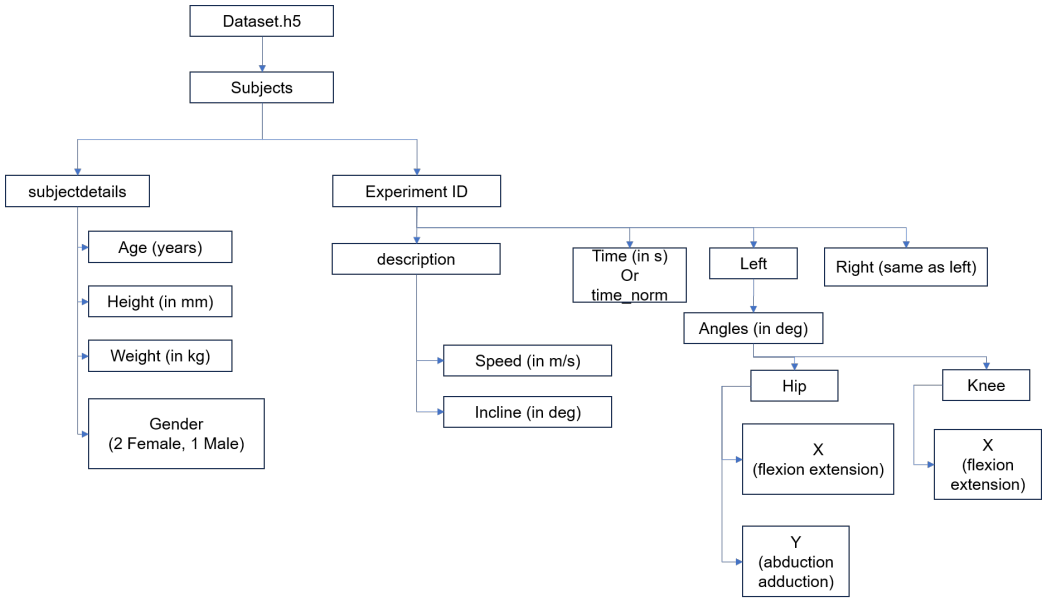

Figure 1: Database structure for the datasetV2.h5 and processed_data.h5 .

4 Python Code

- DatasetsFiltering.py contain the class DatasExtraction with the methods to extract data from the corresponding original databases. The variables at the beginning of the file should be change to the path to the database's file. There is also the function to save the extracted data into an hdf5 file.
  - k_embry_data_extraction for the InclineExperiment folder
  - fukuchi_data_extraction for the WBDSData folder
  - moreira_data_extraction for the MAT files folder
- KeyPointsExtractions.py contain the class KeyPointsExtractions with its principal method key_points_estimation to extract given a trajectory, a time scale, the joint and direction, the key points of the curve. Other methods serve to identify the extremum and inflections points for each specific Degree of Freedom.
- CurveReconstruction.py contain the class CurveReconstruction for the interpolation method used
- DataByKeyPointsFiltering.py have a specific function data_sorting to filter all the data from a file with the structure 1 and extract data for a null incline and which has certain key points that are necessary for the rest
- InterpolationError.py is to compute the RMSE between the real curve and the spline interpolation of its real key points, comparing different degree of interpolation for the spline interpolation
- ExtractDataForTraining has the different functions to extract from the processed_data.h5 the data for the different model for training
- GaitPeriodEstimationModelTraining.py and IntentRecognitionModelTraining.py are files to train respectively the Gait Period Estimation Model and the Gait phase estimation model. They work on the principle of GridSearch for Gaussian Process Regression. The scaler and model ared saved on the computer using Joblib.
- create_model.py is where the function to create a customizable RNN, by choosing all the hyperparameters of the NN at each creation. With that, the GridSearch will used this function and varies the values of those hyperparameters

- ModelTraining.py, the file for training the NN, using GridSearch to find the best combination of hyperparameters. The function partial from functools is used to create a partial function from the one in the create_model.py by fixing some parameters, here the input shape and the output shape.
- ModelEvaluation.py have for objective to evaluate the different errors between generated curves (generated key points + Interpolation) and the real curves, as the errors between the gait percentage of the generated key points and the real key points.

5 Simulation Models

In the simulation models folder, there are 3 types of files / folders:

- The NN models in the folder
- The scalers, which are the .pkl files with scaler in their names, which are used to scale the input data for its respective model
- the GPR files, which are the .pkl files without the name scaler

Example on how to load a regression model:

```
import tensorflow as tf
import pickle
from sklearn.pipeline import Pipeline

# Replace by the name of the regression model
model_path = '
    my_regression_model_hip_abduction_all_population_new'

loaded_model = tf.keras.models.load_model(model_path)

# Replace by the appropriate scaler
scaler_path = '
    saved_scaler_hip_abduction_all_population_new.pkl'
# Load the scaler from the file
with open(scaler_path, 'rb') as f:
    loaded_scaler = pickle.load(f)
print("Scaler loaded successfully.")

pipeline = Pipeline([
('standardizer', loaded_scaler),
('model', loaded_model)
])
```

```
21
22      # X the input data
23      y_predict = pipeline.predict(X)
```

Example on how to load a GPR:

```
1       import tensorflow as tf
2       import joblib
3
4       def custom_optimizer(obj_func, initial_theta, bounds):
5       def objective_function(theta):
6           value = obj_func(theta)
7           # Ensure the function returns a scalar
8           if isinstance(value, (list, np.ndarray)):
9               value = value[0]
10          elif np.isnan(value[0]) or np.isinf(value[0]):
11              return np.inf
12          return value[0]
13      result = minimize(objective_function, initial_theta,
            method='L-BFGS-B', bounds=bounds, options={'maxiter':
            10000})
14      return result.x, result.fun
15
16      standardizer_transformed = joblib.load('
            scaler_intent_recognition.pkl')
17      print("Standardizer␣transformed␣Load!")
18
19      gpr_model_non_transformed = joblib.load('
            best_gp_model_intent_recognition_non_transformed.pkl')
20      print("Gaussian␣Process␣Regressor␣non␣transformed␣Load!")
21
22      # Create the pipeline
23      pipeline = Pipeline([
24      ('standardizer', standardizer_non_transformed),
25      ('gpr', gpr_model_non_transformed)
26      ])
27
28      # X the input data
29      y_predict = pipeline.predict(X)
```

6 Bibliography

[1] Claudiane A. Fukuchi, Reginaldo K. Fukuchi, and Marcos Duarte. "A public dataset of overground and treadmill walking kinematics and kinetics in healthy individuals". In: *PeerJ* 6 (2018), e4640. ISSN: 2167-8359. DOI: 10.7717/peerj.4640.

[2] K. Embry et al. *The Effect of Walking Incline and Speed on Human Leg Kinematics, Kinetics, and EMG*. 2018. DOI: 10.21227/GK32-E868.

[3] Luís Moreira et al. "Lower limb kinematic, kinetic, and EMG data from young healthy humans during walking at controlled speeds". In: *Scientific Data* 8.1 (2021), p. 103. ISSN: 2052-4463. DOI: 10.1038/s41597-021-00881-3. URL: https://www.nature.com/articles/s41597-021-00881-3.