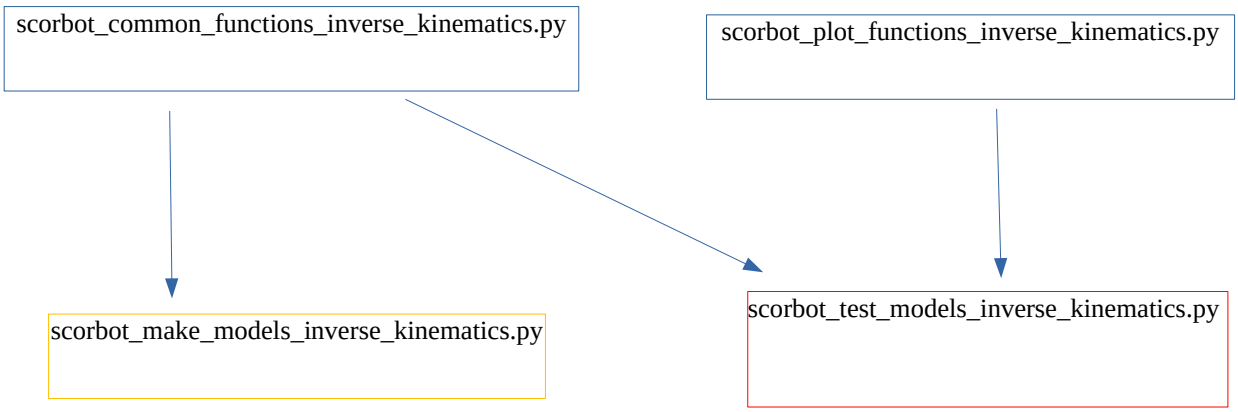
File description



1) scorbot_common_functions_inverse_kinematics.py:

Scorbot direct kinematic equations: **Xe**, **Ye**, **Ze**, **Op**, **Oy**
Auxiliary functions:

Euclidean_distance: two formats: single X,Y.Z coordinates or point1 and point2 coordinates.

data_set_creation (samples, directory, training_data_file): if the dataSet does not exists, creates de dataMat structure. Otherwise, if reads the .csv file and creates dataMat.

2) scorbot_make_models_inverse_kinematics.py:

Needs *data_set_creation* scorbot_common_functions_inverse_kinematics.py

build_and_train_general_model(model_name,dataMat): here is where the MLP-ANN is builded, trained and saved.

build_and_train_pitch_model(model_name,dataMat): this is a function that produce MLP-ANN models to predict the pitch and yaw of the Scorbot based on the X,Y,Z coordinates.

3) scorbot_plot_functions_inverse_kinematics.py:

angles_scatter_plot (dataMat,model,directory,model_name_prefix):
angles_hexbin_plot (dataMat,model,directory,model_name_prefix):
density_error_plot (samples, error, mean, directory, model_name_prefix):
histogram_and_density_error_plot (samples, error, mean, directory, model_name_prefix):
histogram_error_plot (samples, error, mean, directory, model_name_prefix):
bar_error_matrix_plot (error_matrix, Last_model,directory):
bar_error_matrix_plot2 (error_matrix, Last_model,directory):
threeD_error_plot (error, dataMat, directory, model_name_prefix):
threeD_error_plot_first_quadrant (error, dataMat, directory, model_name_prefix):

4) scorbot_test_models_inverse_kinematics.py:

load_models (directory, model_name_prefix, model_number): the directory with the .h5 MLP-ANN models should exist and have the models inside. These models should be created before by the “scorbot_make_models_inverse_kinematics.py” script.

get_error (end_point, model): it calculates the error between the MLP-ANN output and the inverse kinematic equations output. It uses the *Euclidean_distance* of *scorbot_common_functions_inverse_kinematics.py*.

get_best_solution (end_point, model_list, model_number): it outputs the best solution from a MLP-ANN list. It uses *get_error* function.

get_model_error_vector (test_samples_size, dataMat, model_number, model_list, directory, error_data_file): computes the lowest error (best solution) of a dataset (dataMat) and a list of MLP-ANNs (model_list). It uses *get_best_solution* function.

get_models_error_matrix (test_samples_size, dataMat, model_list, directory, error_data_file): it computes an error matrix. Each column “I” uses 0 to “I” models (MLP-ANN) to get the best error/solution. It also saves the matrix to a file. This function is computationally expensive. It uses *get_best_solution* function.

test_bootstrapping_approach_with_N_models (test_samples_size, dataMat, model_number, model_list, directory): It uses *get_model_error_vector* function. Calls that function for a “model_number” of MLP-ANN (models) included in the “model_list”. It gives some statistics about the errors, and then plots different error graphs. It needs the *scorbot_plot_functions_inverse_kinematics.py* file. This function is used in “main” with 1 model (classical approach) and 50 models (bootstrapping approach). In the first case results and plots are saved in “model_0_results” directory. In case of 50 models results are saved in “model_50_results” directory.

```
# Step 1: get data set (dataMat) and load MLP (models):
#samples_, dataMat = data_set_creation(test_samples_size, directory, test_data_file_name)
model_list=load_models (directory,model_name_prefix,Last_model+1)

# Step 2 (optional): calculate and compare the errors of the whole models.
error_matrix=get_models_error_matrix(test_samples_size_error_matrix, dataMat, model_list, directory, error_data_file_error_matrix)
bar_error_matrix_plot (error_matrix, Last_model, directory)

 # Step 3: get results from the classical MLP that uses one model (one neural network).
 test_bootstrapping_approach_with_N_models (test_samples_size, dataMat, 0, model_list, directory_model_0)

 # Step 4: get results from the bootstrapping approach that takes N models
# (neural networks) instead one 50 in this case). The objective is to compare
# with results of Step 3.
test_bootstrapping_approach_with_N_models (test_samples_size, dataMat, 50, model_list, directory_model_50)
```