TFMESH

TFMESH (pronounced TF-Mesh) is a package for **T**rajectory **F**using, **M**otion **E**stimation and **S**top **H**andling.

This package performs the following tasks:

- Basic I/O
  - load trajectory in a close-to-NGSIM format
  - save trajectory data in NGSIM format
- RTS Smoother
  - perform trajectory fusing if multiple observations are available
  - produce smoothed position data
  - produce estimated velocity and acceleration
- RANSAC stop detection and handling
  - perform stop detection
  - perform stop handling, including spline interpolation to connect moving and stopped parts

Getting started

System requirement

Packaged required in this project is listed in `requirements.txt`. It is recommended to use Anaconda distribution as most packages are already self-contained. The only exception is `filterpy` which can be installed by running `pip install filterpy`.

The code is developed using Python 3.8.3, but should be working on most modern Python releases.

Example and tutorial

To begin, start by looking `main.ipynb`, which contains the step-by-step procedures from importing a raw vehicle trajectory data to smoothed trajectory with motion data.

This package TFMESH uses two well established methods: an RTS smoother for "TFME" and a RANSAC based detector for "SH". For an introduction to RTS smoother, including its simplier version of Kalman filter, please see https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python/. For tutorial on RANSAC, please see https://www.cse.psu.edu/~rtc12/CSE486/lecture15.pdf.

Input data format

The input data takes a format very close to NGSIM data, except some data columns are added to facilitate trajectory fusing.

As shown in `libStep1.py` under `encode_veh_ud()`, the following are column number (counting from 0) where the key information are located from input data. This can be updated to reflect specific needs.

```
IND_POS = 5         # column for raw position
IND_FID = 1         # column for frame ID
IND_CAM_ID = 2      # 2 if for NGSIM data from Lizhe, 18 if for 3D LiDAR data
IND_LANE_ID = 13    # column for lane ID
IND_US_FLAG = 21    # column for upstream and downstream indicator
```

Function hierarchy

As used in `main.ipynb`

- libFileio.py
    - get_veh
    - load_data
    - save_data_step1
    - save_data_step2
    - save_data
- libStep1.py
    - combine_cam_motion_est_ud
        - encode_veh_ud
        - est_init_v
        - init_ca
        - measurement_noise_model
- libStep2.py
    - ns_and_s_handle
        - index_true_region
        - index_stop_region
            - index_true_region
        - spline_near_stop