# SEMTA Post-Processing Server

## Overview

* The SEMTA system uses a remote server to collect data and perform post-processing tasks such as data fusion and tracking.
* This system is implemented in two Python scripts. Data processing is performed by Tracking.py, while Server.py implements an HTTP server using Flask and Dash.
* Processed data can be viewed using the web interface, which can be found at the server’s web address, followed by ‘/dashboard’. For example, ‘http://localhost:5000/dashboard’.

## Server Deployment

* A Docker container using Docker Compose has been configured for installation & deployment to any target operating system:
  + Install Docker and Docker Compose using preferred package manager
  + Set desired port in PostProcessing/docker-compose.yml, defaults to 5000
  + Issue the following commands within the “PostProcessing” directory:
    - docker-compose build
    - docker-compose up -d

## Web Interface

* This server implements a simple web interface using Plotly and Dash.
* Web interface can be reached at ‘[SERVER IP ADDRESS:PORT]/dashboard’.
* Includes menu to modify tracking filter parameters and repeat processing
* Plots on web interface can be adjusted and saved to local system.
* See example web interface readout in Figure 1.
* Tracking filter parameters are described below:
  + Maximum Target Speed – Approximate maximum potential target speed, used for initializing target velocity variance.
  + Maximum Target Acceleration – Approximate maximum potential target acceleration, used for initializing target acceleration variance.
  + Cross-Track/Along-Track Motion Variance – Tuned parameter of Kalman filter which weights the motion matrix of the nearly-constant-acceleration model.

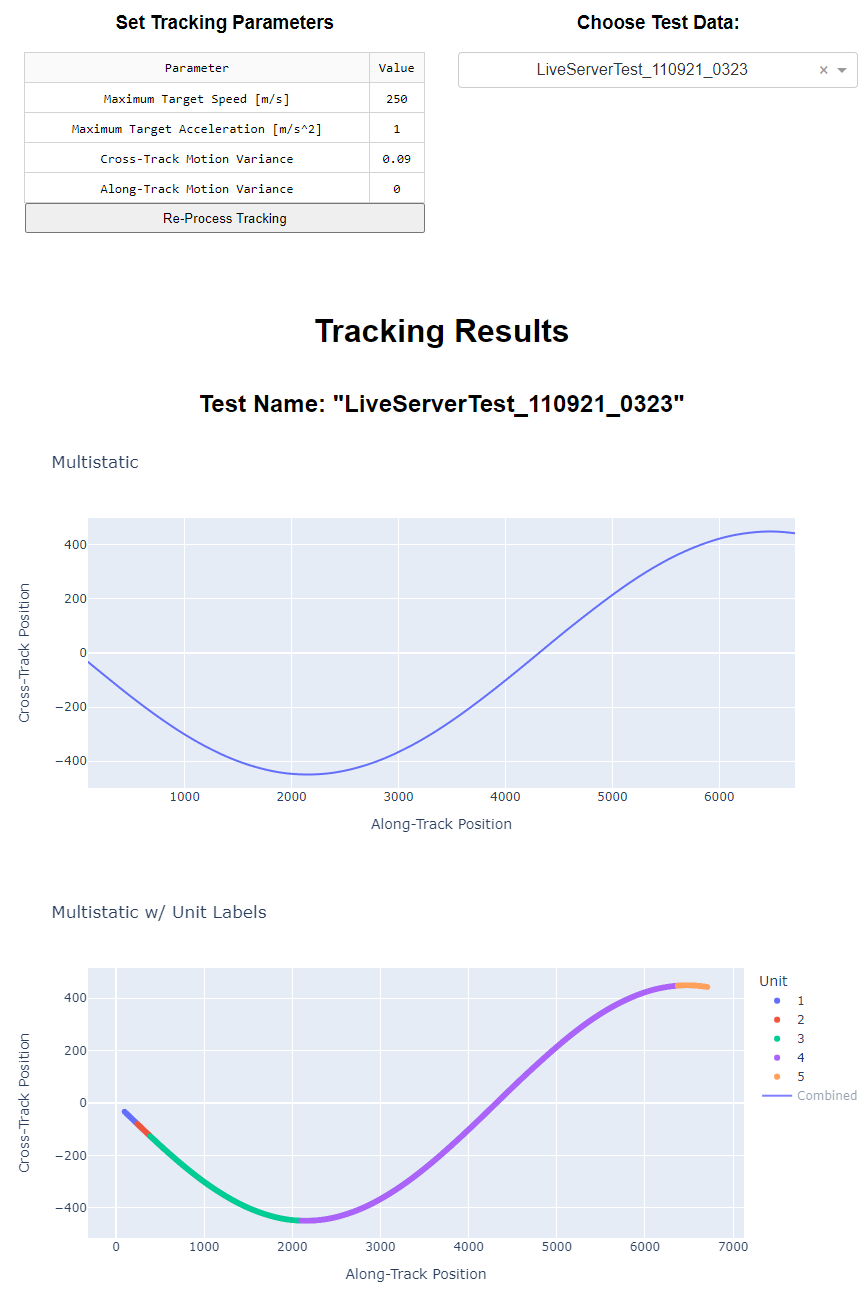


Figure 1. Example of web interface

## Client Usage

* Radar data is currently packaged in a .mat file, for use with MATLAB simulation and processing results. This format should be modified for file format of final use case.
  + The .mat file contains the following fields:
    - ‘n\_fr’
      * Number of frames of measurement data
      * Integer value
    - ‘range’
      * Estimated range to target, in meters
      * Floating point vector of length ‘n\_fr’
    - ‘vel’
      * Estimated Doppler range rate of target, in meters per second
      * Floating point vector of length ‘n\_fr’
    - ‘az’
      * Estimated azimuth bearing of target determined by monopulse algorithm, in degrees
      * Floating point vector of length ‘n\_fr’
    - ‘steer’
      * Antenna array beamsteering angle, in degrees
      * Floating point vector of length ‘n\_fr’
    - ‘SNR’
      * Estimated signal-to-noise ratio of target reflection, in decibels
      * Floating point vector of length ‘n\_fr’
    - ‘time’
      * Timestamps of measurements
      * Floating point vector of length ‘n\_fr’
    - ‘radar\_pos’
      * Radar position in cartesian coordinates, in meters
      * Floating point vector of length 3, describing down-range x cross-range x vertical position
* Radar measurement data is uploaded to server using HTTP POST request.
  + Default URI is ‘/’, for example http://127.0.0.1:5000/
  + File is attached to request body using ‘form-data’ format, with key “file”
* If file is successfully sent, it will be saved in the “PostProcessing/Input” folder.
  + A folder within “PostProcessing/Output” is generated using the name of the file.
  + .csv files of tracked target coordinates, along with .png files of resulting scatter plots, are saved to the new output file.
  + One of each type of output file is created for each single-unit tracking result, plus one for the multistatic results.
* A browser or REST API client can be used to verify connection and correct operation.
  + GET request to root URI of ‘/’ should return HTTP status 200 if connected successfully

## Open Issues

* Tracking.py file will need to be modified to accommodate file type used as output of radar system.
  + .mat file is currently used, as it is directly output from MATLAB.
  + File unpacking code is found in lines 552-573 of Tracking.py.
* HTTP server and port will need to be configured and exposed to WAN.
  + HTTP server currently runs on localhost:5000.
  + Port 5000 can be modified by editing Dockerfile and rebuilding Docker image.