Design:

Classes:

Train:

All required APIs are provided in class Train. Constructor of Train takes a filename that contains all datasets and computes in background.

Train contains an instance of Computation, which takes care of all computation and a model, which stores all data, including raw input data and estimate data.

Computation:

It is an interface for all computation models. The only required methods in the interface is

addData(Data d), which adds a new data to computation and update the internal states. Computation takes a model and updates it.

All implementations of Computation uses KalmanFilter to update and predict. Computation1 uses a normal KalmanFilter which models the process as a constant velocity movement without changing directions. Computation2-4 uses extended Kalman Filter to take care of changing of directions. We use Computation 4 in our final design.

Computation also provides some method to analyze how well the prediction is.

Model:

where we manage data. Data are stored by arraylists of Data. Model provides APIs to modify or get data. Model is mutable, but encapsulated. All getters in model either return an immutable object or a copy of the original object.

Data:

A single entry of data. Contains x, y, time and a boolean valid to indicate whether the data is valid or not. Field valid is used in computation where we filter out invalid data. Data is immutable with all fields public final. This makes data storage safe.

Parse:

Parse data from input file into a two dimensional array.

Filter:

We implemented two filters: KalmanFilter and Smoother. Smoother uses exponential smoothing algorithm, but it did not work very well, so we use Kalman filter in our final design.