Localization with Noisy Android Raw GNSS Measurements

Xu Weng, KV Ling

School of Electrical and Electronic Engineering

Nanyang Technological University

October 11, 2023

- High-performance GNSS Equipment:

PwrPak7-E1 from HEXAGON

VEXXIS® GNSS-800 Series Antennas from HEXAGON

EVK-M10QSAM from ublox

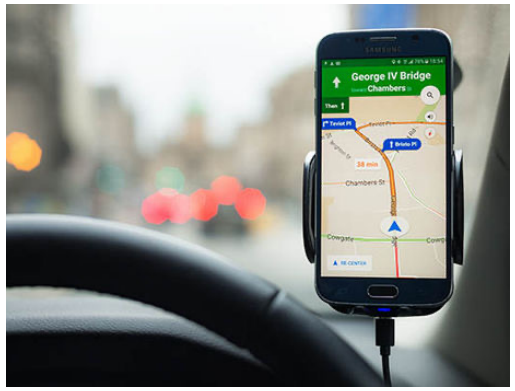
PolaRx5S from Septentrio

- Inaccessibility in daily life
  - *Heavy*
  - *Large*
  - *Expensive*

Background

- By contrast, smartphones are:
  - *Affordable*
  - *Portable.*

- Android raw GNSS measurements are expected to enable various exciting localization-based applications.



Navigation

Bike Sharing

Emergency Location Service

Mobile AR

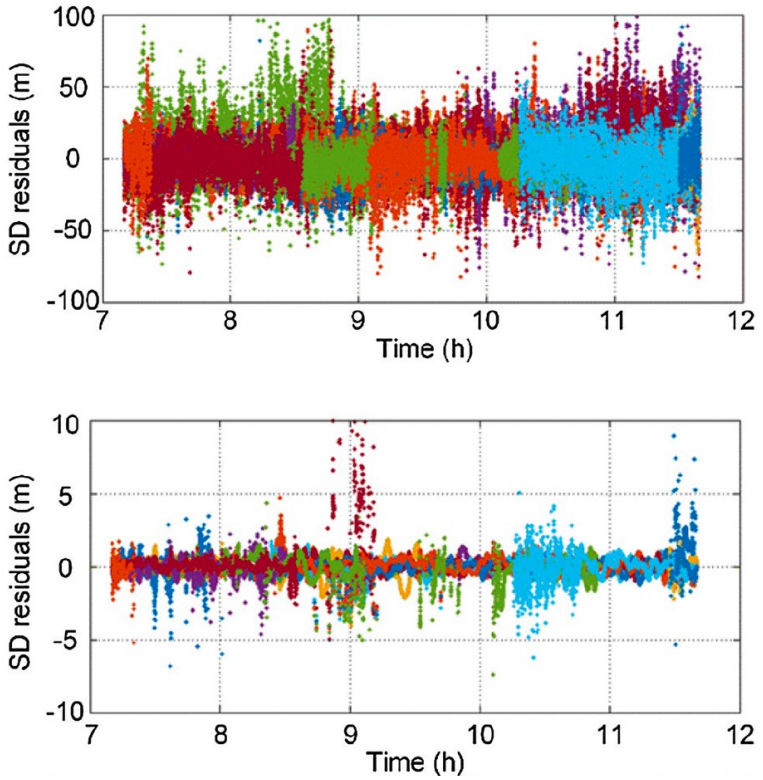
(From Google Images)

Challenges

- Low-cost chips and antennas of smartphones results in:
  - ***Noisy measurements***
  - Pseudorange noise of phones is about ***an order of*** magnitude larger than geodetic-quality measurements [1].

Nexus 9 (Android Phone)

NovAtel (Geodetic-quality Receiver)



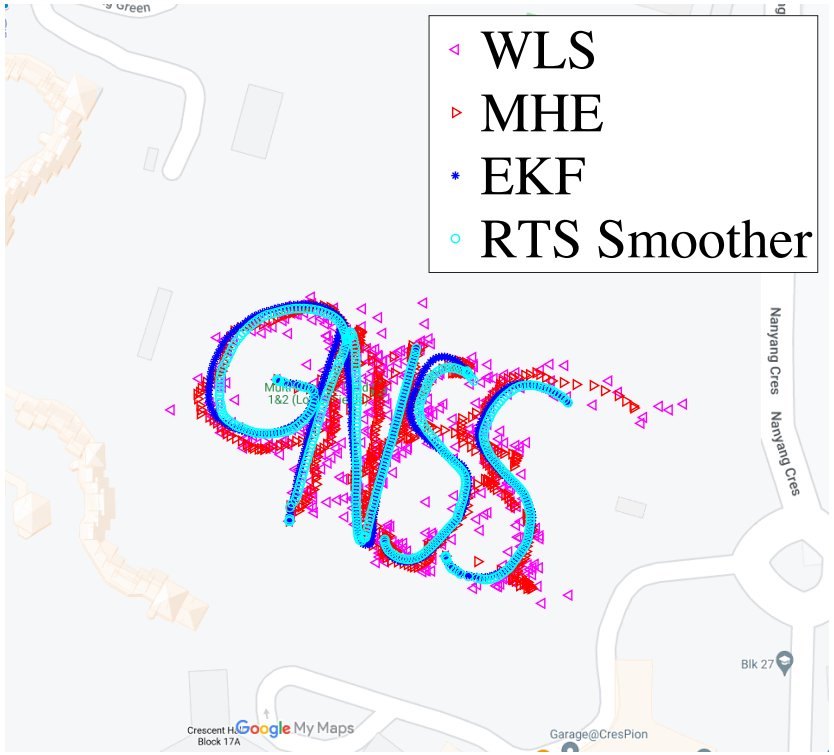
Pseudorange residuals collected from smartphones and specialized receivers[1]

[1] Zhang, X., Tao, X., Zhu, F., Shi, X., & Wang, F. (2018). Quality assessment of GNSS observations from an Android N smartphone and positioning performance analysis using time-differenced filtering approach. *Gps Solutions*, *22*, 1-11.

Motivation

- Denoise Android raw GNSS measurements:
  - ➢Filtering or smoothing theories are well-established.
  - ➢But *their implementation on Android phones are seldom mentioned*.

- We present systematic engineering details about adapting filtering and smoothing techniques to Android raw GNSS measurements.
  - ➢Pseudorange based position, velocity, and time (PVT) calculation;
  - ➢ Weighted Least Squares (WLS) method;
  - ➢ Moving Horizon Estimation (MHE);
  - ➢ Extended Kalman Filter (EKF);
  - ➢ Rauch-Tung-Striebel (RTS) smoother
  - ➢Our codes are open-source!



Data was collected by Xiaomi Redmi K60

The Pipeline of Android Raw GNSS Data Collection and Processing

GNSS Satellite

GNSS Signals

Android Smartphone

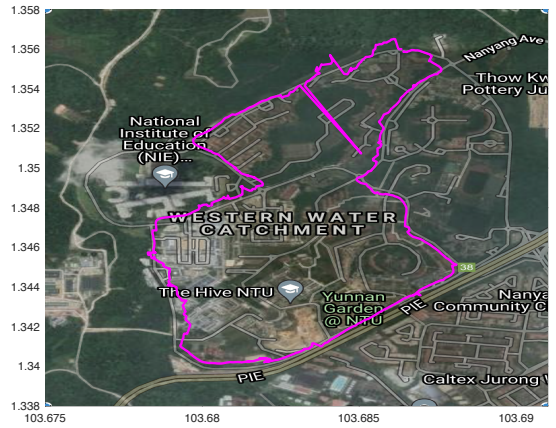
GnssLogger App
Developed with Google

GnssLogger

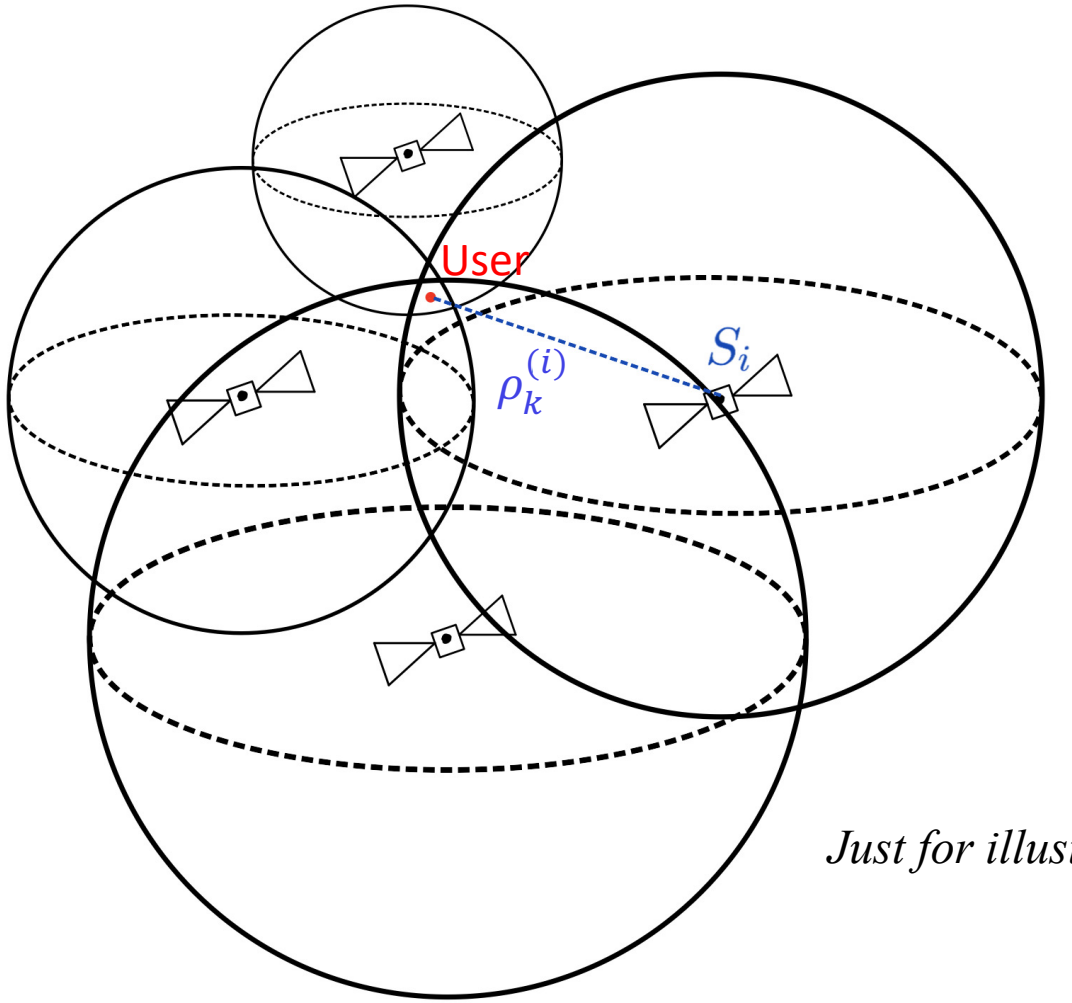
Log GNSS Data

GNSS Data File

Analysis Software



Preliminaries of GNSS (1/3)



Just for illustration:

- For *M* visible satellites, we can establish a nonlinear equation system where
  - Knowns are
    - Satellite locations $(x_k^{(i)}, y_k^{(i)}, z_k^{(i)})$,
    - Ranges measurements between satellites and users (pseudoranges) $\rho_k^{(i)}$,
  - Unknowns are
    - *User locations* (x_k, y_k, z_k) .

$$\sqrt{\left(x_k - x_k^{(1)}\right)^2 + \left(y_k - y_k^{(1)}\right)^2 + \left(z_k - z_k^{(1)}\right)^2} + \delta t_{u_k} = \rho_k^{(1)}$$

$$\sqrt{\left(x_k - x_k^{(2)}\right)^2 + \left(y_k - y_k^{(2)}\right)^2 + \left(z_k - z_k^{(2)}\right)^2} + \delta t_{u_k} = \rho_k^{(2)}$$

$$\vdots$$

$$\sqrt{\left(x_k - x_k^{(M)}\right)^2 + \left(y_k - y_k^{(M)}\right)^2 + \left(z_k - z_k^{(M)}\right)^2} + \delta t_{u_k} = \rho_k^{(M)}$$

- Assemble pseudoranges with Android raw GNSS measurements

- Clock Field
  - 'TimeNanos'
  - 'TimeUncertaintyNanos'
  - 'LeapSecond'
  - 'FullBiasNanos'
  - ……
- Measurement Field
  - 'Cn0DbHz'
  - 'ConstellationType'
  - 'MultipathIndicator'
  - 'PseudorangeRateMetersPerSecond'
  - ……

Android Raw GNSS Measurements

The n^{th} satellite at the k^{th} time step:

Geometry distance | User's clock bias | Pseudorange noise

Corrected Pseudorange

$$\rho_{c_k}^{(n)} = r_k^{(n)} + \delta t_{u_k} + \varepsilon_k^{(n)}$$

Geometry distance rate | User's clock drift | Pseudorange Rate noise

Corrected Pseudorange Rate

$$\dot{\rho}_{c_k}^{(n)} = \dot{r}_k^{(n)} + \delta f_{u_k} + \dot{\varepsilon}_k^{(n)}$$

- Position, Velocity, and Time (PVT) estimation using the WLS algorithm
  - Basically, it is a ***Gauss-Newton*** algorithm:

$$\begin{aligned}\mathbf{X}_k &= \tilde{\mathbf{X}}_k + \Delta\mathbf{X}_k \\ &= \tilde{\mathbf{X}}_k + \left(\mathbf{W}_k\mathbf{G}_k\right)^{+}\mathbf{W}_k\mathbf{b}_k\end{aligned}$$

Weight Matrix

Approximate state

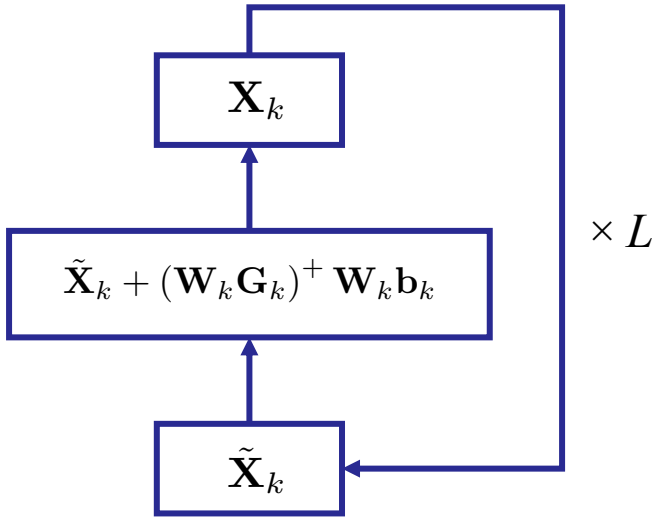
Jacobian Matrix

Residual Vector

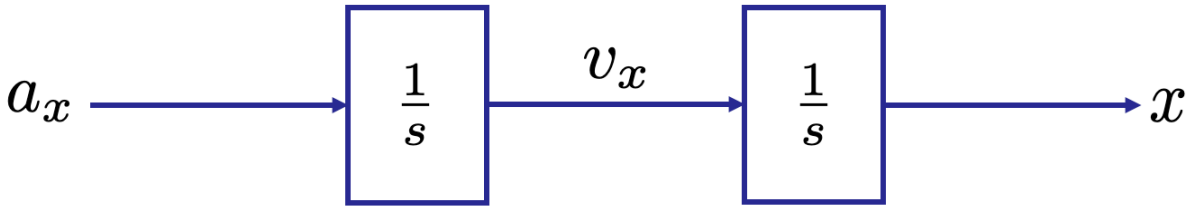
where the unknown state to be estimated is

$$\mathbf{X}_k = [x_k, v_{x_k}, y_k, v_{y_k}, z_k, v_{z_k}, \delta t_{u_k}, \delta f_{u_k}]^T$$

Position *Velocity* *Clock Bias and Drift*



Filtering and Smoothing: Process Models of Android Phones



Dynamic modal of the phone's location and velocity on the *x* axis

$$\mathbf{X}_{x_k} = \mathbf{A}^{(x)}_{t_k,t_{k-1}} \mathbf{X}_{x_{k-1}} + \mathbf{W}_{x_{k-1}}$$

where

$$\mathbf{X}_{x_k} = [x_k, v_x]^T$$

$$\mathbf{A}^{(x)}_{t_k,t_{k-1}} = \begin{bmatrix} 1 & T_{s_k} \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{Q}_{x_{k-1}} = \mathrm{E}\left(\mathbf{W}_{x_{k-1}} \mathbf{W}^T_{x_{k-1}}\right) = \begin{bmatrix} \frac{1}{3} S_{v_x} T^3_{s_k} & \frac{1}{2} S_{v_x} T^2_{s_k} \\ \frac{1}{2} S_{v_x} T^2_{s_k} & S_{v_x} T_{s_k} \end{bmatrix}$$

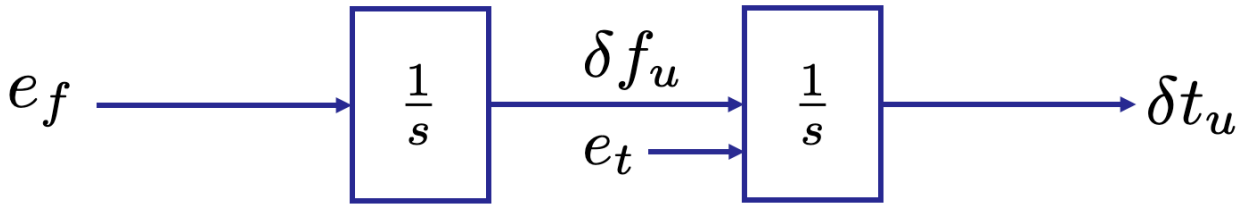
$$S_{v_x} = \mathrm{E}(a_x^2).$$

★

estimated velocities at previous epochs

$$S_{v_x} = \mathrm{E}(a_x^2) \approx a_x^2 \approx \left(\frac{\hat{v}_{x_{k-1}} - \hat{v}_{x_{k-2}}}{T_{s_{k-1}}} \right)^2$$

Filtering and Smoothing: Process Models of Android Phones



Dynamic modal of the phone's location and velocity on the *x* axis

$$\mathbf{X}_{t_k} = \mathbf{A}^{(t)}_{t_k,t_{k-1}} \mathbf{X}_{t_{k-1}} + \mathbf{W}_{t_{k-1}}$$

where

$$\mathbf{X}_t = [\delta t_u, \delta f_u]^T$$

$$\mathbf{A}^{(t)}_{t_k,t_{k-1}} = \begin{bmatrix} 1 & T_{s_k} \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{Q}_{t_{k-1}} = \mathrm{E}\left(\mathbf{W}_{t_{k-1}} \mathbf{W}^T_{t_{k-1}}\right) = \begin{bmatrix} S_t T_{s_k} + \frac{1}{3} S_f T^3_{s_k} & \frac{1}{2} S_f T^2_{s_k} \\ \frac{1}{2} S_f T^2_{s_k} & S_f T_{s_k} \end{bmatrix}$$

$$S_t = \mathrm{E}(e_t^2),\ S_f = \mathrm{E}(e_f^2).$$

★

estimated clock bias at previous epochs

$$S_t = \mathrm{E}(e_t^2) \approx e_t^2 \approx \left(\frac{\delta\hat{t}_{u_{k-1}} - \delta\hat{t}_{u_{k-2}}}{T_{s_{k-1}}} - \delta\hat{f}_{u_{k-1}} \right)^2$$

$$S_f = \mathrm{E}(e_f^2) \approx e_f^2 \approx \left(\frac{\delta\hat{f}_{u_{k-1}} - \delta\hat{f}_{u_{k-2}}}{T_{s_{k-1}}} \right)^2$$

estimated clock drift at previous epochs

Filtering and Smoothing: Moving Horizon Estimation

- Thus, the following linear discrete-time system can describe the state of Android smartphones with low dynamics:

$$\mathbf{X}_k = \mathbf{A}_{k,k-1}\mathbf{X}_{k-1} + \mathbf{W}_{k-1}$$

$$\mathbf{b}_k = \mathbf{C}_k(\mathbf{X}_k - \tilde{\mathbf{X}}_k^-) + \mathbf{E}_k$$

where

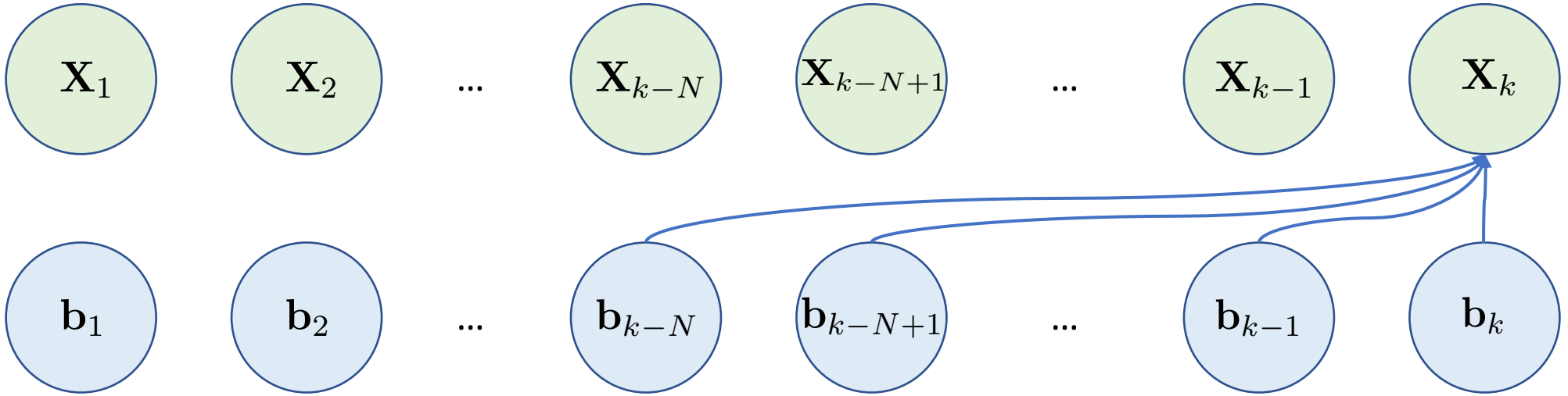
$$\mathbf{C}_k = \mathbf{G}_k$$

$$\mathbf{E}_k = [\varepsilon_k^{(1)}, \dot{\varepsilon}_k^{(1)}, \varepsilon_k^{(2)}, \dot{\varepsilon}_k^{(2)}, \cdots, \varepsilon_k^{(M)}, \dot{\varepsilon}_k^{(M)}]^T.$$

Pseudorange and pseudorange rate errors

Filtering and Smoothing (1/3)

- Moving Horizon Estimation (MHE):



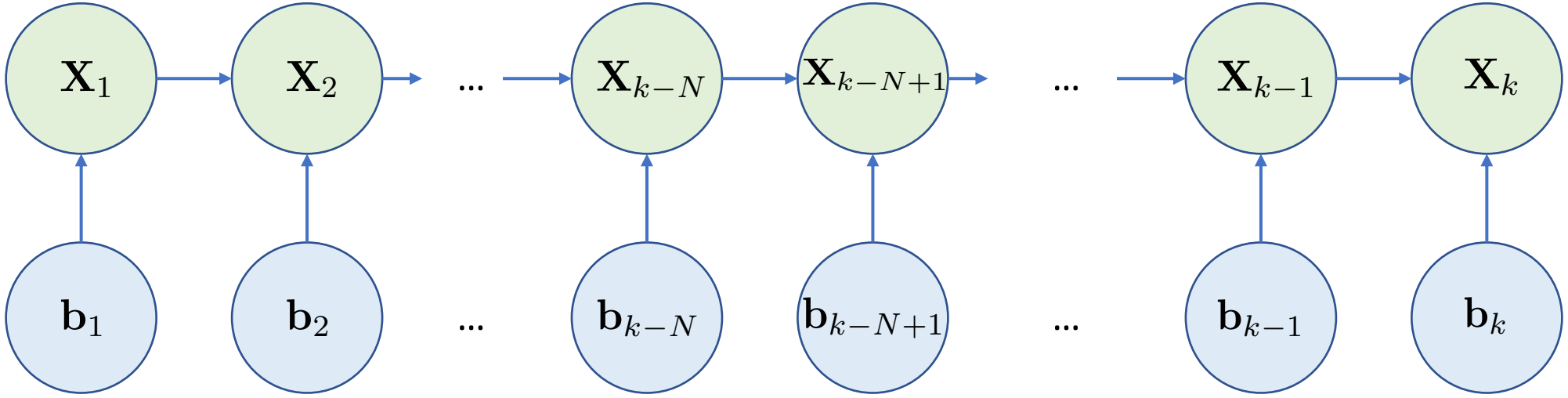
$$\hat{\mathbf{X}}_k = \mathbf{M}_N^{+}\mathbf{Y}_{k,N} + \tilde{\mathbf{X}}_k$$

where

$$\mathbf{Y}_{k,N} = \left[\mathbf{b}_k^T, \mathbf{b}_{k-1}^T, \cdots, \mathbf{b}_{k-N}^T\right]^T$$

$$\mathbf{M}_N = \left[(\mathbf{C}_k)^T, (\mathbf{C}_{k-1}\mathbf{A}_{k,k-1}^{-1})^T, \cdots, (\mathbf{C}_{k-N}\mathbf{A}_{k-N+1,k-N}^{-1}\cdots\mathbf{A}_{k-1,k-2}^{-1}\mathbf{A}_{k,k-1}^{-1})^T\right]^T$$

- Extended Kalman Filter (EKF):



$$\hat{\mathbf{X}}_k^- = \mathbf{A}_{k,k-1}\hat{\mathbf{X}}_{k-1}$$
$$\mathbf{P}_k^- = \mathbf{A}_{k,k-1}\mathbf{P}_{k-1}\mathbf{A}_{k,k-1}^T + \mathbf{Q}_{k-1}$$
$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{C}_k^T\left(\mathbf{C}_k\mathbf{P}_k^-\mathbf{C}_k^T + \mathbf{R}_k\right)^{-1}$$
$$\hat{\mathbf{X}}_k = \hat{\mathbf{X}}_k^- + \mathbf{K}_k\mathbf{b}_k$$
$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k\mathbf{C}_k\right)\mathbf{P}_k^-$$

★

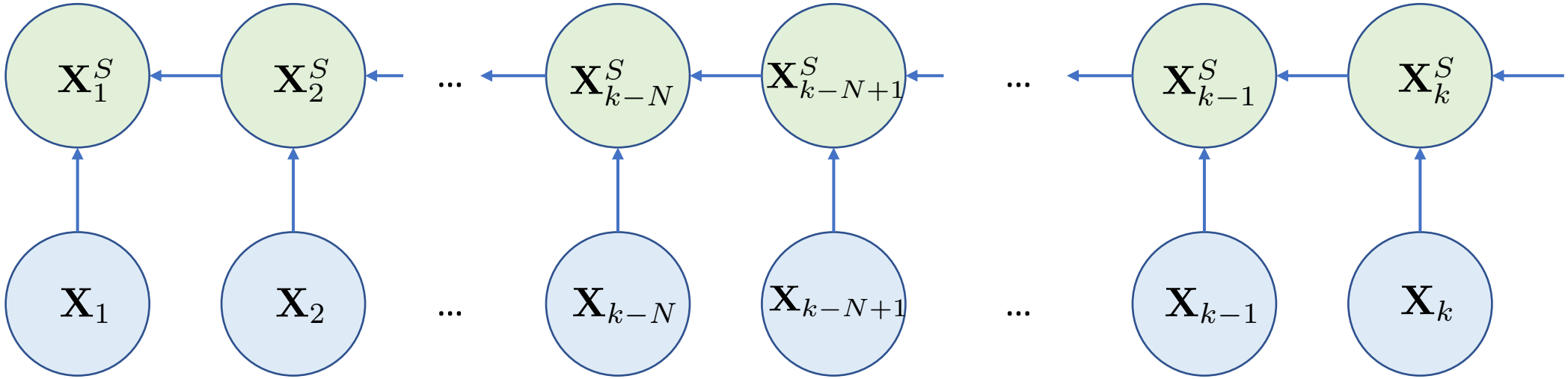
$$\mathbf{R}_k = \mathrm{E}\left(\mathbf{E}_k\mathbf{E}_k^T\right) = \begin{bmatrix} \sigma_{\rho_k}^{(1)^2} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \sigma_{\dot{\rho}_k}^{(1)^2} & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \sigma_{\rho_k}^{(M)^2} & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \sigma_{\dot{\rho}_k}^{(M)^2} \end{bmatrix}$$

$$\sigma_{\rho_k}^{(n)^2} = \mathrm{E}\left(\varepsilon_k^{(n)^2}\right) = ReceivedSvTimeUncertaintyNanos^2$$

$$\sigma_{\dot{\rho}_k}^{(n)^2} = \mathrm{E}\left(\dot{\varepsilon}_k^{(n)^2}\right) = PseudorangeRateUncertaintyMetersPerSecond^2$$

Filtering and Smoothing (3/3)

- Rauch-Tung-Striebel (RTS) Smoother:



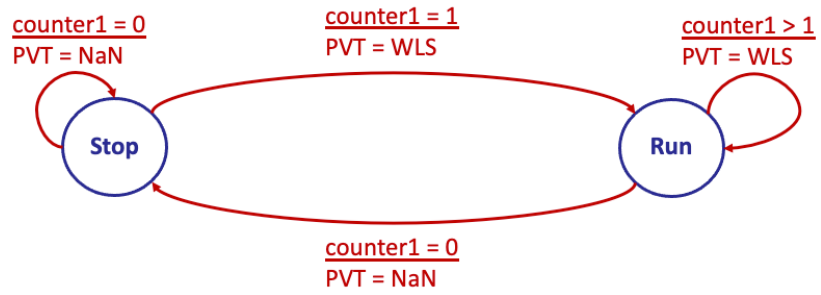
$$\hat{\mathbf{X}}_k^S = \hat{\mathbf{X}}_k + \mathbf{S}_k\left(\hat{\mathbf{X}}_{k+1}^S - \hat{\mathbf{X}}_{k+1}^-\right)$$

$$\mathbf{P}_k^S = \mathbf{P}_k + \mathbf{S}_k\left(\mathbf{P}_{k+1}^S - \mathbf{P}_{k+1}^-\right)\mathbf{S}_k^T$$

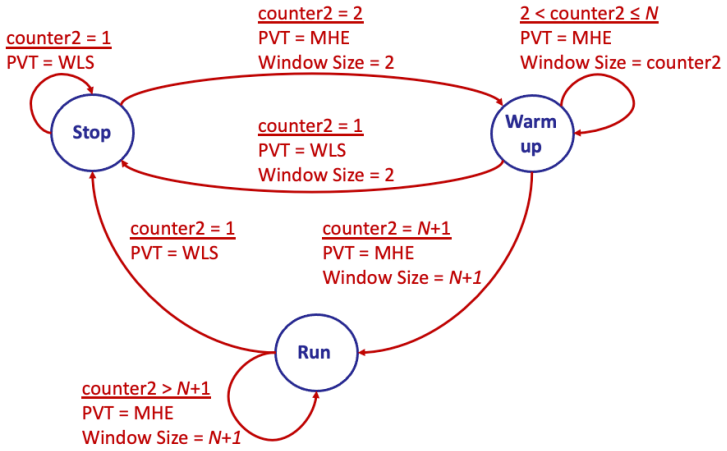
$$\mathbf{S}_k = \mathbf{P}_k\mathbf{A}_{k,k-1}\left(\mathbf{P}_{k+1}^-\right)^{-1}$$

Practical Implementation for Discontinuous Data

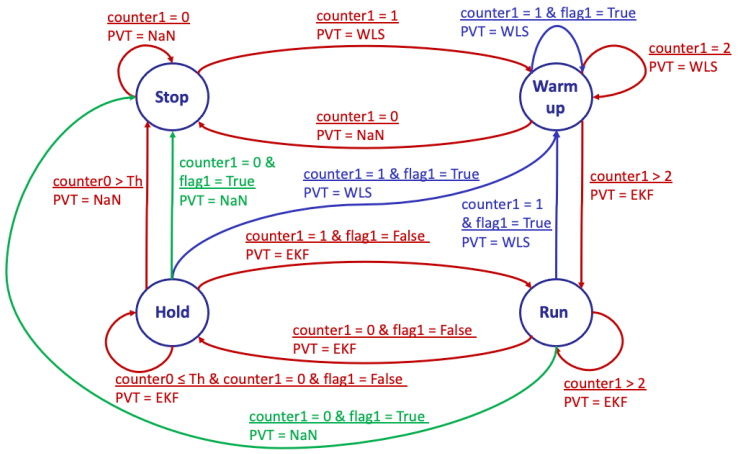
- We use finite state machine (FSM) to handle the discontinuity of Android raw GNSS measurements:
  - Satellite discontinuity
    - Localization fails due to fewer than four available satellites.
  - Clock discontinuity
    - The timestamps are not continuous for two adjacent measurements.
  - Pseudorange discontinuity
    - The pseudorange change between two consecutive epochs is much larger than the expected value.



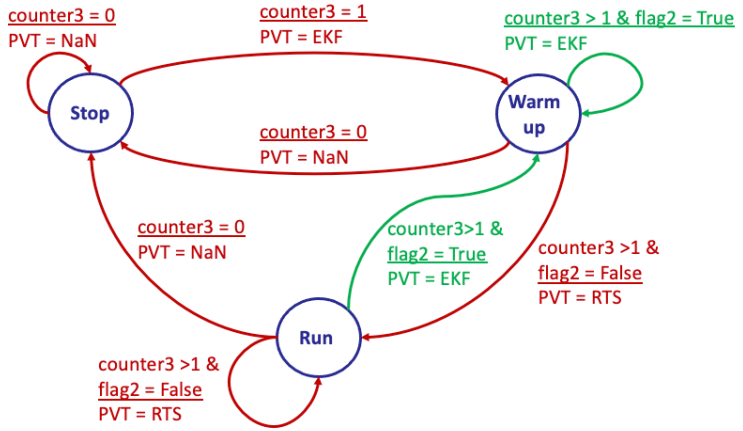
FSM for the WLS algorithm



FSM for MHE

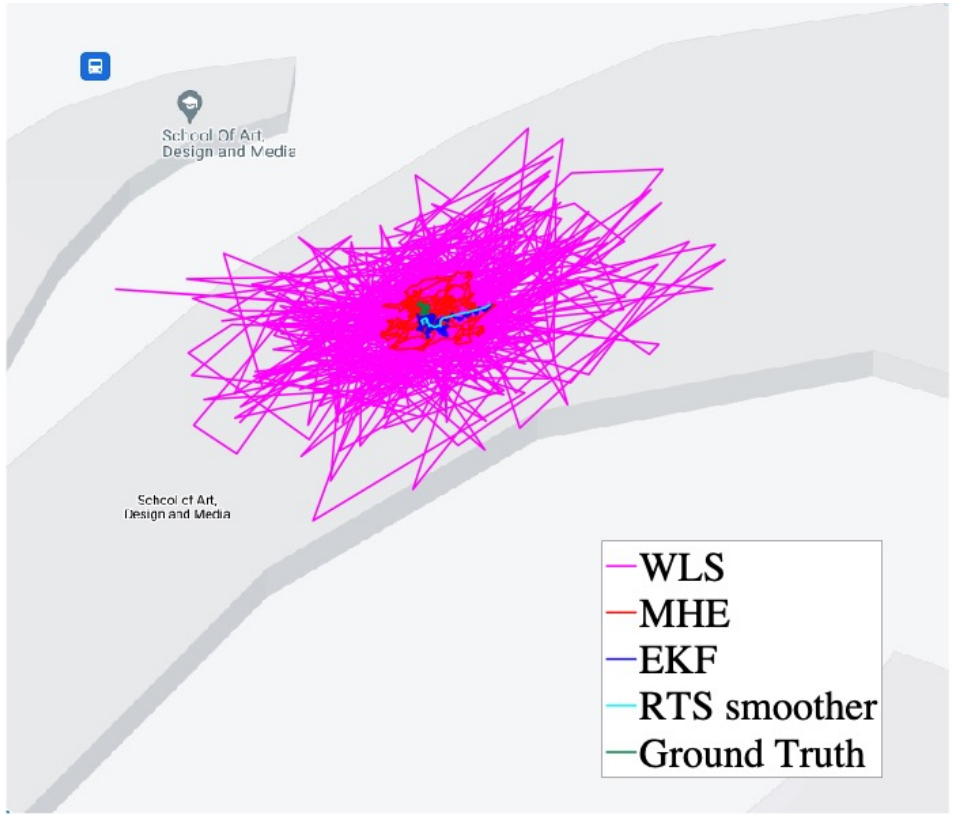


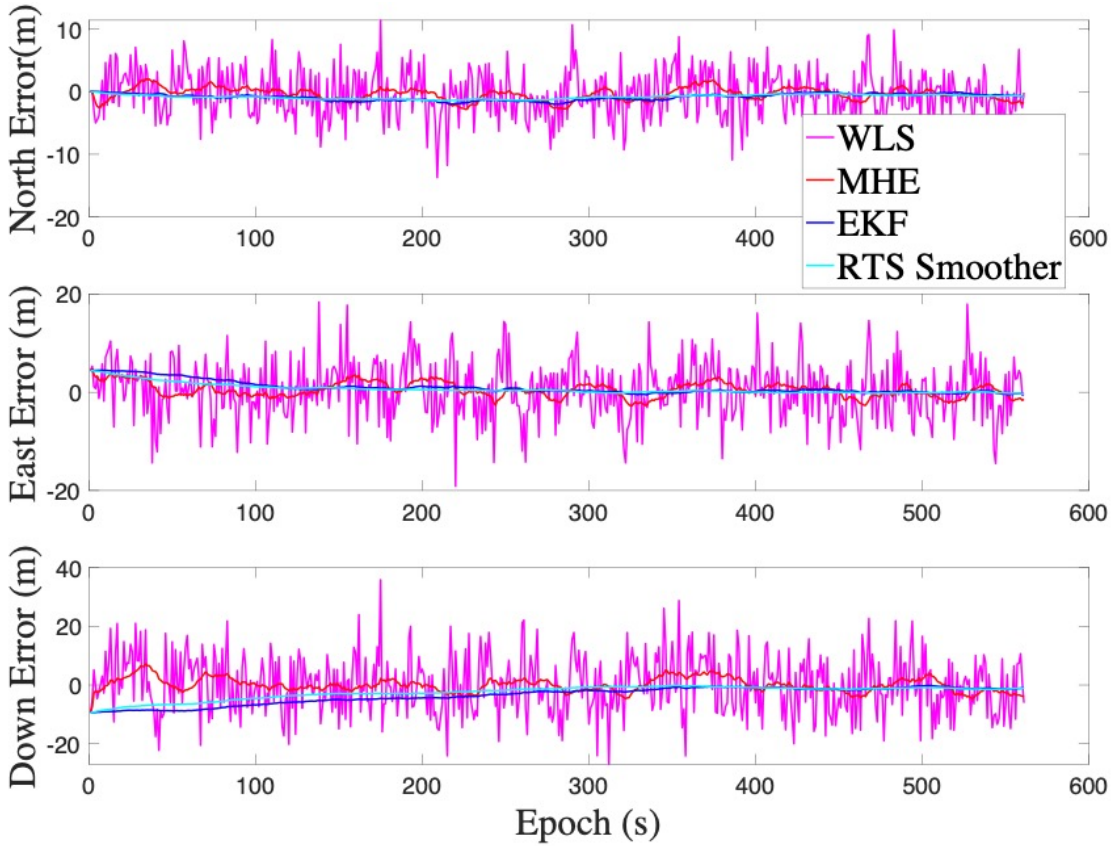
FSM for EKF



FSM for RTS smoother

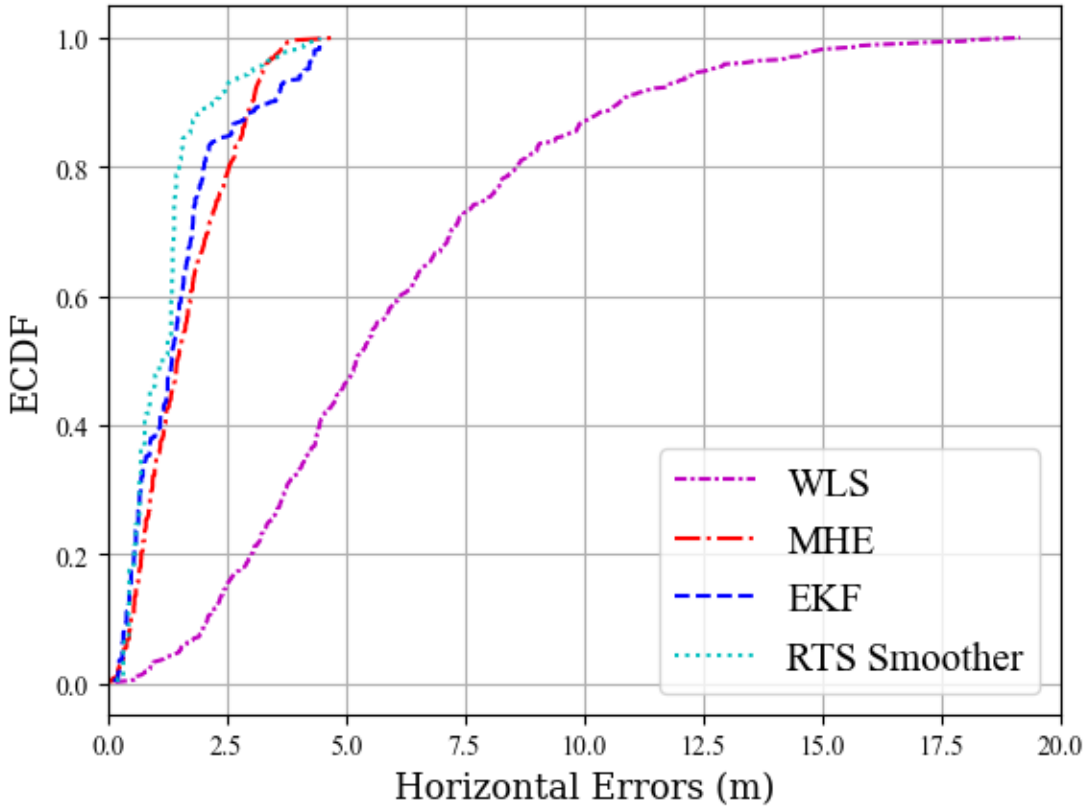
Experiments and Results: Static Scenario

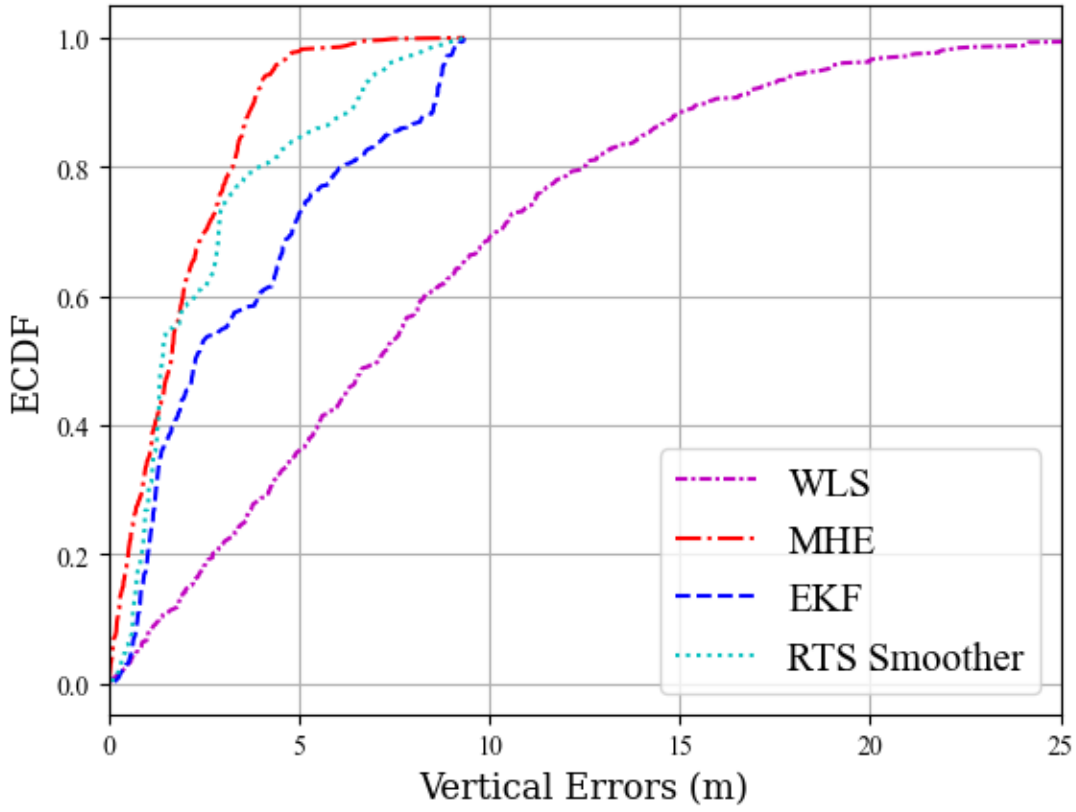




Android data was collected by *Huawei Mate 10 Pro* on the roof of NTU ADM;
Ground truth data was collected by a u-blox GNSS receiver.

Experiments and Results: Static Scenario

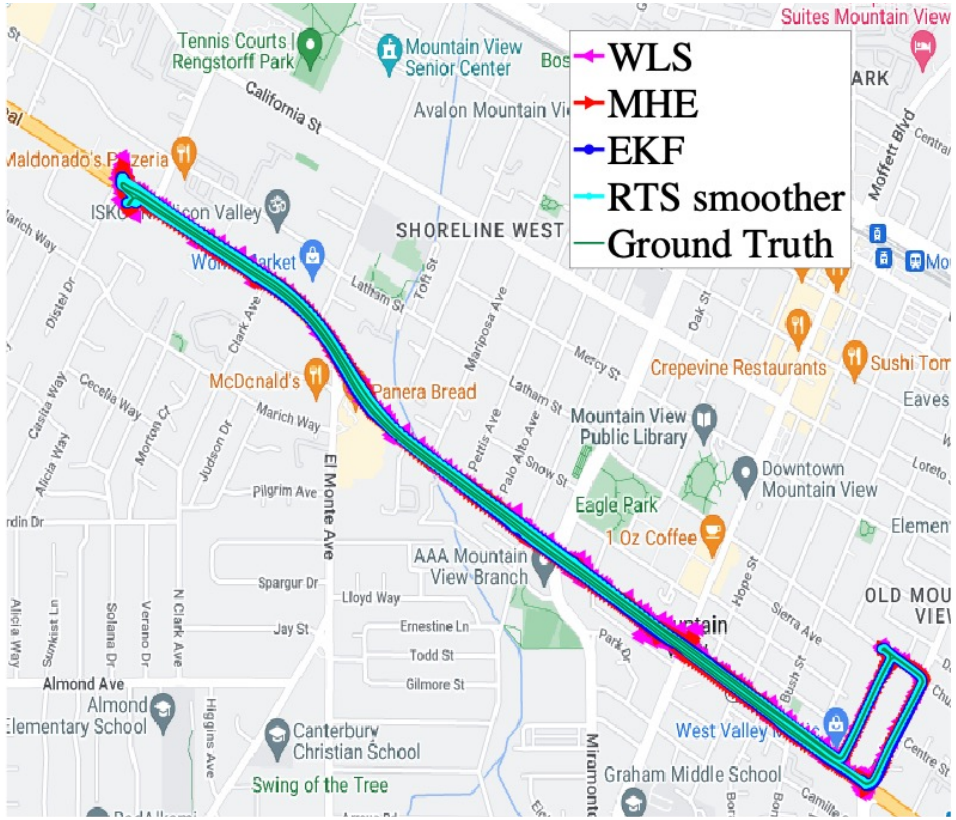


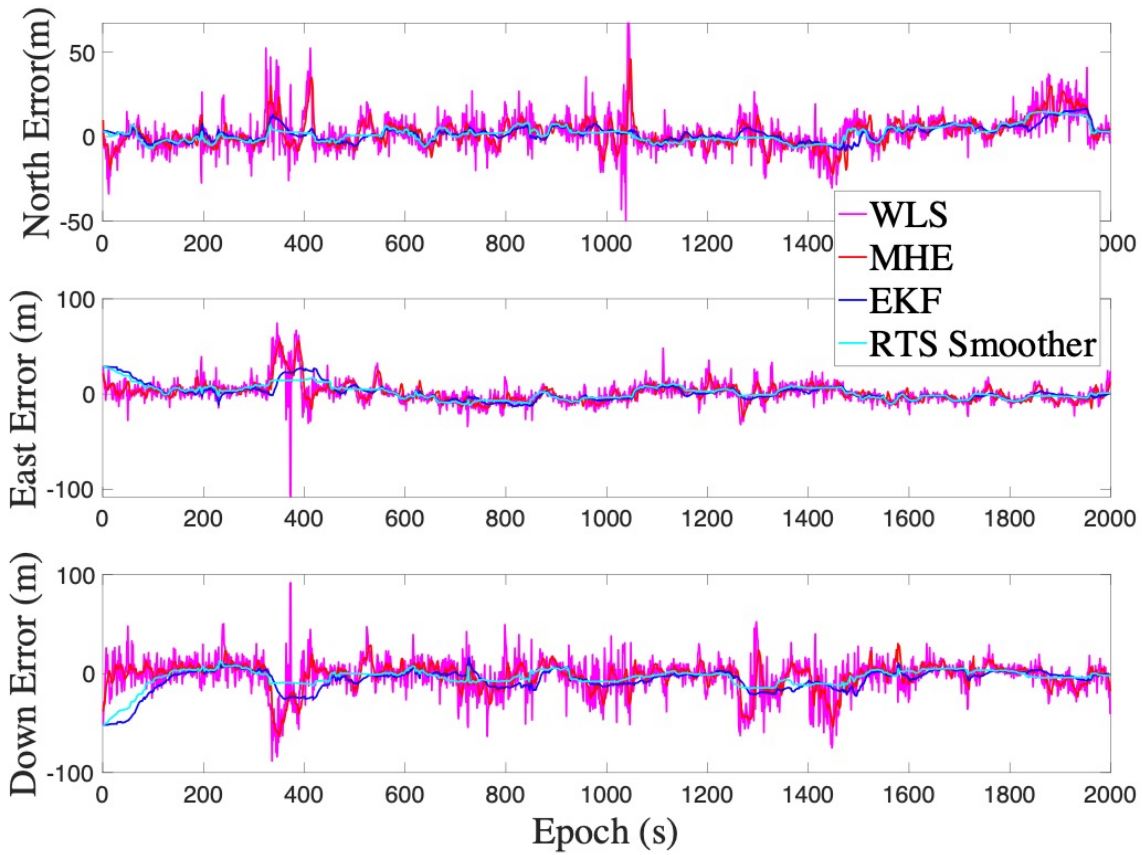


Methods	Horizontal Score* (meter)↓	Improvement↑
WLS	8.9280	-
MHE	2.3671	73.4%
EKF	2.7485	69.2%
RTS Smoother	**2.1051**	**76.4%**

*Means of the 50th and 95th percentile of horizontal distance errors computed with Vincenty's formulae

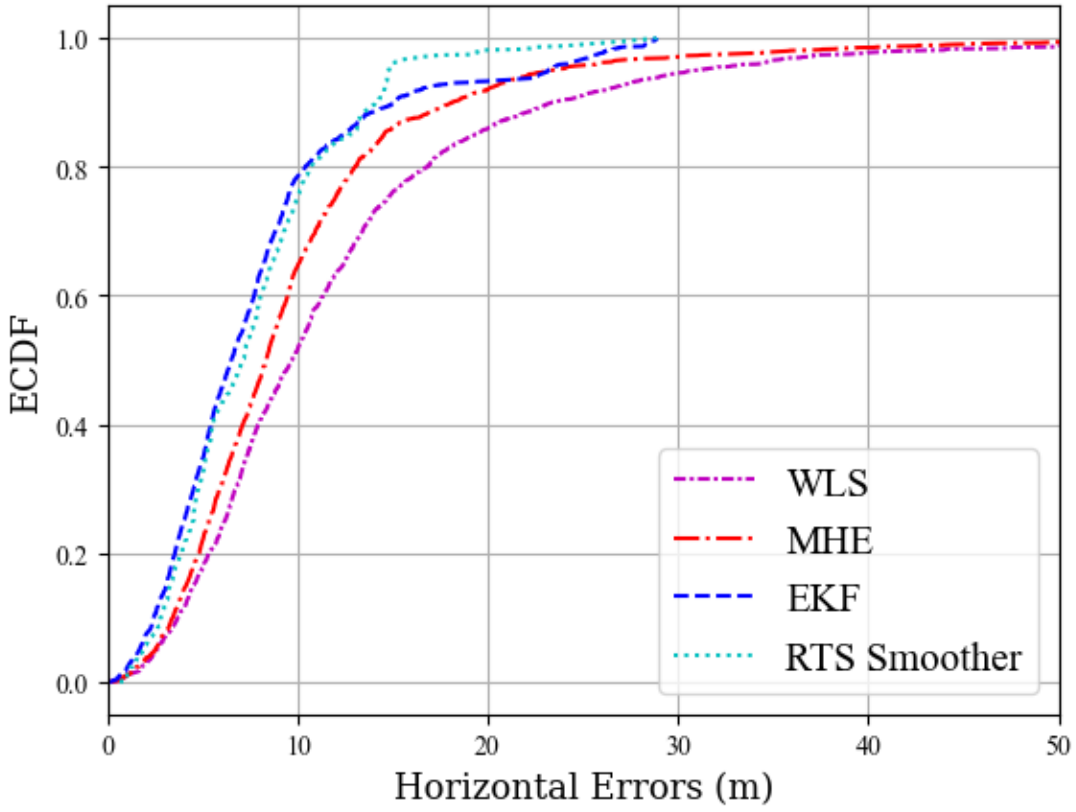
Experiments and Results: Dynamic Scenario

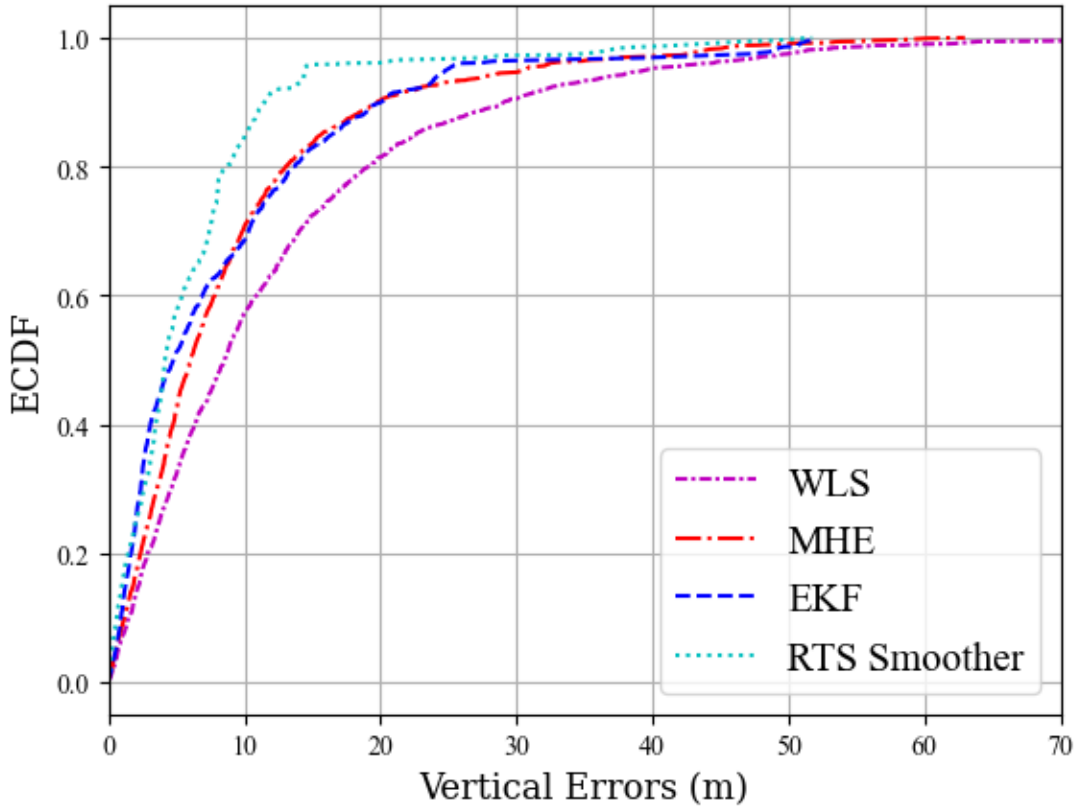




Android data was collected by *Pixel 4* at Mountain View by Google;
Ground truth data was collected by a NovAtel SPAN system.

Experiments and Results: Dynamic Scenario

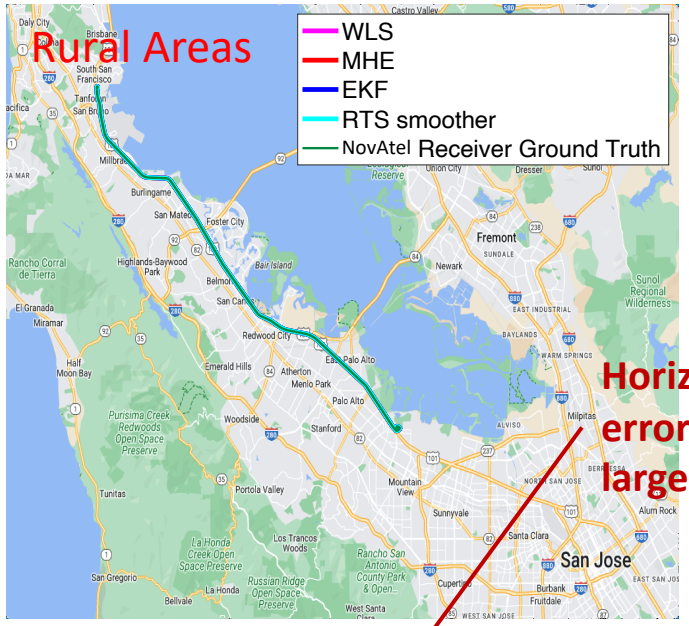


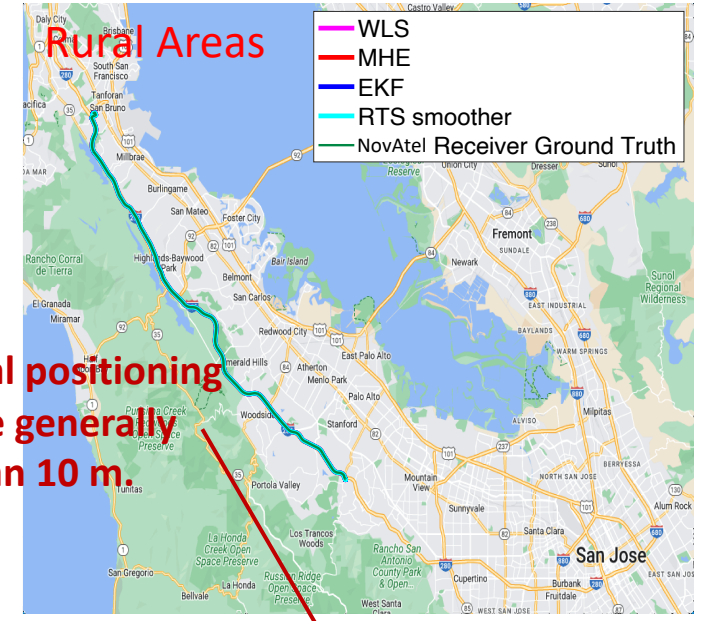


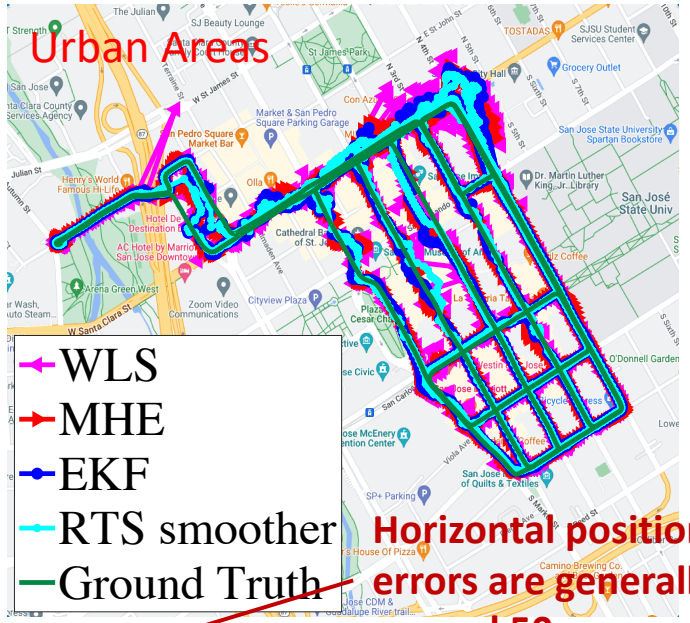
Methods	Horizontal Score*(meter)↓	Improvement↑
WLS	20.4651	-
MHE	15.9024	22.3%
EKF	14.8676	27.3%
RTS Smoother	**10.9495**	**46.5%**

*Means of the 50th and 95th percentile of horizontal distance errors computed with Vincenty's formulae

More Results

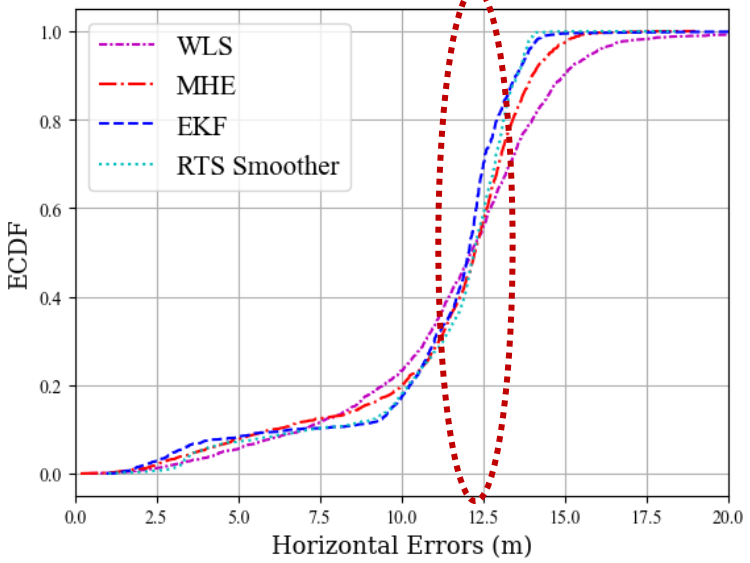


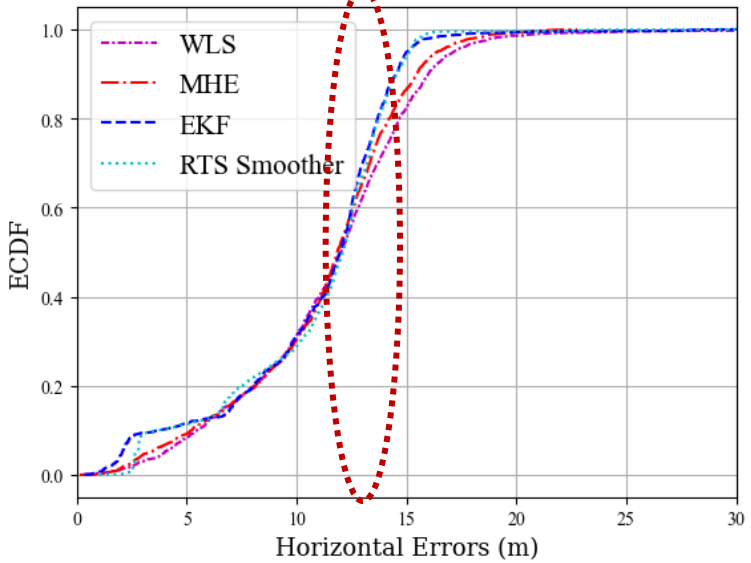


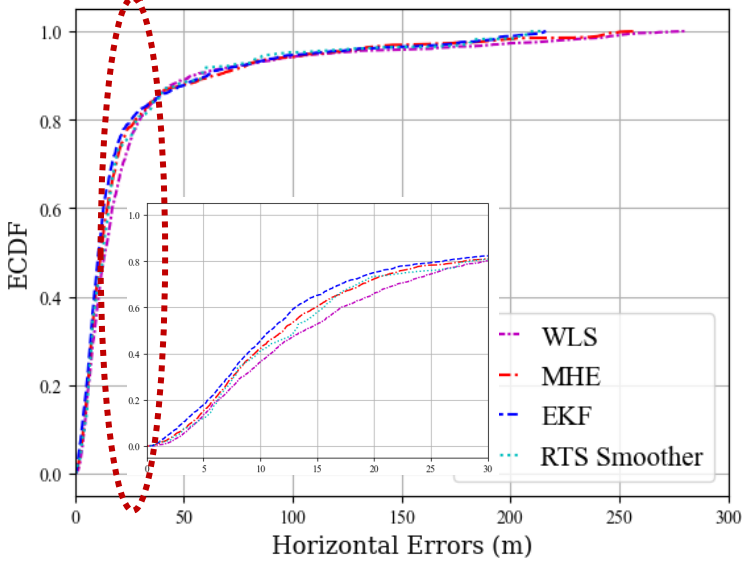


Horizontal positioning errors are generally larger than 10 m.

Horizontal positioning errors are generally around 50 m.

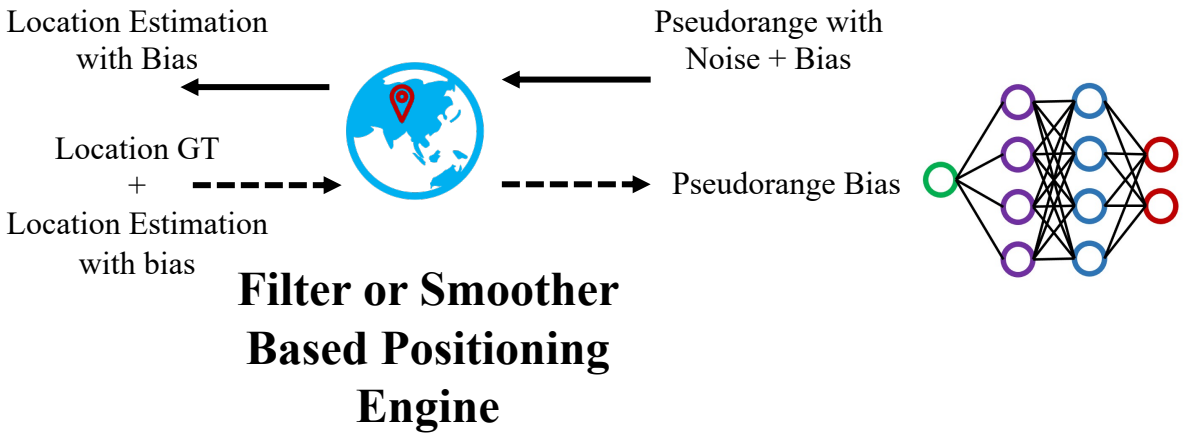






Conclusion and Future Work

- We detail how to calculate position, velocity, and time (PVT) using Android raw GNSS measurements.
- We employ MHE, EKF, and RTS smoother to filter or smooth noisy Android raw GNSS measurements to improve localization performance.



Weng, X., Ling, K. V., & Liu, H. (2023). ***PrNet: A Neural Network for Correcting Pseudoranges to Improve Positioning with Android Raw GNSS Measurements***. arXiv preprint arXiv:2309.12204.

Thank you!