Reference Positioning Engine & Measurements Database for Wi-Fi RTT (FTM) Positioning

Leor Banin, Ofer Bar-Shalom, Nir Dvorecki, and Yuval Amizur

Abstract

The following whitepaper describes a reference positioning engine (PE) and time-delay measurements database that can be used for the development and performance evaluation round-trip time (RTT)-based, Wi-Fi location systems. The database includes fine-timing measurements (FTM) timestamps that were collected in an indoor Wi-Fi venue. As a turnkey for using this database, a reference Matlab® code of a positioning engine (PE) that can be executed by a Wi-Fi client station, is provided. The reference PE demonstrates the usage of the database and enables to analyze the performance of the PE using additional information included in the database files.

Index Terms

Geolocation, Positioning, Position Estimation, Location Estimation, Indoor navigation, Fine timing measurement, FTM, Time delay estimation, WLAN, Wi-Fi, IEEE 802.11

I. Introduction

The following whitepaper outlines the usage guidelines for Matlab® code of a reference positioning engine (PE) that uses time delay measurements for estimating and tracking the position of a client receiver. The receiver is assumed to be equipped with IEEE802.11 WLAN (Wi-Fi) chipset that is capable of fine-timing measurements [1]- [4]. To test the PE's performance, the Matlab code is provided along with a set of files that contain time delay measurements collected in various indoor venues. The files contain both real-life, time delay timestamps of

L. Banin, O. Bar-Shalom, N. Dvorecki and Y. Amizur are with Intel's Location Core Division, 94 Em Hamoshavot Rd., Petah Tikva 49527, Israel. leor.banin@intel.com, ofer.bar-shalom@intel.com, nir.dvorecki@intel.com, yuval.amizur@intel.com

Wi-Fi units operating in the venue that were collected by a client that was roaming within the venue. Additional files, containing synthetic, noisy range measurements. The measurement files consist of all the information required for enabling the analysis of PE performance and the accuracy of its position estimates, (for which "ground truth" (reference) position information of the client receiver is included). The Wi-Fi network deployed in the venue consists of fine timing measurements (FTM) responders, which are capable of running FTM sessions with client devices to measure the client's range to the responder. The client feeds its range measurements into a positioning engine, based on an extended Kalman filter (EKF), which produces an estimate of its current location and tracks it.

The remainder of this paper is organized as follows. Section II describes the contents of the software package, its installation and its usage. Section III contains a description of the code and implementation notes. Finally, Section IV provides some numerical examples of the performance measures that can be obtained by executing the PE code on the given inputs.

II. Software Download, Installation & Usage

The following section describes the how to install and execute the simulation environment.

A. Download Information

The package may be downloaded from the Intel's repository on the GitHub website:
https://github.com/intel/wifi-location-core-pe-and-measurement-database.
To install the software, simply create a folder on your PC's local drive and copy all the files into it. The folder should contain the following files:

B. Matlab Code Files

1) `mainFtmPE.m` - contains the main function to run the simulation environment.
2) `testFtmPeConfig.m` - contains a function for extracting the test configuration.
3) `ReadFtmMeasFile.m` - contains a function for reading the `*.csv` measurement files.
4) `RunFtmPE.m` - a function that implements the main simulation loop: it loops over the entire measurements list and executes the EKF per each measurement.
5) `KfFtmclass.m` - contains a class of methods implementing the EKF.
6) `PlotFtmPeResults.m` - contains a function that implements the results visualization.

C. *Indoor Time-Delay Measurements Database*

The reference code of the FTM PE is distributed along with several measurements files. These files enable the code users to evaluate the performance of the reference PE and additional PEs that can accept the measurement format. The measurements files included are listed next. Additional files may be added in the future. The data files can be found in a `data` folder, which contains multiple sub-folders, each containing several files (including the measurement files themselves).

1) Indoor Time-Delay Database Contents: The data folders currently contained in the database are the following:

1) Office1_80MHz
2) Office2_80MHz
3) Office3_80MHz
4) Office4_80MHz
5) Office5_80MHz
6) Office6_40MHz

Each folder contains 4 files:

1) `<data folder name>.csv` - the real measurements data (including GT and RSPs locations, as aforementioned)
2) `<data folder name>_noisySynthRanges.csv` - the synthesized measurements data (including GT and RSPs locations, as aforementioned)
3) `<data folder name>_RSP_LIST.csv` - a table containing the FTM responders' locations. This data is used by the data visualization function.
4) `<data folder name>_VenueFile.mat` - a `*.mat` file containing the venue architectural map and additional venue-related parameters. This data is used by the data visualization function.

2) Indoor Time-Delay Measurements File Format: Each file is a `*.csv` file format that spreads over 9 columns (A-I), where each line represents a measurement with the following data format:

- Column A - timestamp (measured in seconds)
- Column B - ground-truth (GT) x-coordinate (measured in meters in local coordinates system)

- Column C - GT y-coordinate (measured in meters in local coordinates system)
- Column D - GT z-coordinate (measured in meters in local coordinates system)
- Column E - Measured range to the responder (measured in meters)
- Column F - Responder (RSP) x-coordinate (measured in meters in local coordinates system)
- Column G - RSP y-coordinate (measured in meters in local coordinates system)
- Column H - RSP z-coordinate (measured in meters in local coordinates system)
- Column I - RSP index

D. Software Execution

The code was tested on Matlab® R2016b (but may also run on its earlier versions). It has no toolbox dependencies other than than basic Matlab. To execute the code:

1) Set current Matlab® to the folder to which the files were extracted (or add that folder to the Matlab® search path).

2) In the Matlab® Command Window type: `mainFtmPE('<data folder name>')` or `mainFtmPE()`

The following options are presently supported:

- `mainFtmPE()` - executes the FTM PE on the data contained in, `'Office1_80MHz'`.
- `mainFtmPE('<data folder name>')` - executes the FTM PE on the data contained in `<data folder name>`.

III. Code Usage & Implementation Notes

A. Code Configuration

The function `testFtmPeConfig.m` may be used for configuring various parameters and behaviors of the code. The listing of the code is as follows.

```
function cfg = testFtmPeConfig(sessionFolder)
if  ~nargin
    cfg.sessionFolder = 'Office1_80MHz';
else
    cfg.sessionFolder = sessionFolder;
end
cfg.name                    = cfg.sessionFolder;
cfg.measFile                = [cfg.name,'.csv'];
cfg.rPosFile                = [cfg.name,'_RSP_LIST.csv'];
cfg.VenueFile               = [cfg.name,'_VenueFile.mat'];
%**************************************************************************
cfg.UseSyntheticMeas        = 1; % 1 = Use synthetic measurements
                                 % 0 = Use real measured data
if cfg.UseSyntheticMeas
    cfg.measFile = [cfg.name,'_noisySynthRanges.csv'];
    cfg.name = cfg.measFile(1:end-4);
end
%**************************************************************************
cfg.Rsp2remove              = [];% set e.g., cfg.Rsp2remove = [1,6] to remove RSPs {1,6}
cfg.scaleSigmaForBigRange = 1; % 1 = Enable STD scaling for range, otherwise set to 0
cfg.outlierFilterEnable     = 1; % 1 = Enable Outlier Filtering, otherwise set to 0

cfg.MaxRangeFilter          = 50;% filter out ranges above this threshold
cfg.OutlierRangeFilter      = 30;% enable outlier range filtering above this threshold
cfg.gainLimit               = 3; % EKF gain limit
%**************************************************************************

cfg.knownZ                  = 1.4;% Known client height [meter]
cfg.rangeMeasNoiseStd       = 1.0;% Range measurement noise Std [meter].
cfg.zMeasNoiseStd           = 0.1;% Height measurement noise Std [meter].

cfg.posLatStd               = 1.0; % Q - sys. noise [meter per second];
cfg.heightStd               = 0.1; % Q - sys. noise [meter per second];
cfg.biasStd                 = 0.01;% Q - sys. noise [meter per second];
cfg.init.posLatStd          = 1;   % P - state cov. [meter]
cfg.init.heightStd          = 0.2; % P - state cov. [meter]
cfg.init.biasStd            = 0.5; % P - state cov. [meter]


% measurement type defines
cfg.MEAS_RANG               = 1;
cfg.MEAS_CONST_Z            = 2;
end
```

Listing 1. The `testFtmPeConfig.m` Function Listing

1) Special settings: The code supports the following configurations:

1) `cfg.UseSyntheticMeas` - enables to choose whether the code will be fed using real measured ranges data (subjected to additive receiver noise, non-line of sight (NLoS) biases, channel impairments etc.), or will be fed using synthesized range measurements, which consist of the true geometrical range between the actual (ground-truth) location of the client receiver and the specific FTM responder with additive Gaussian noise with 0 mean and standard deviation of 1 meter. As the EKF is optimized for Gaussian-distributed errors, the synthesized measurements may be used for validating the EKF performance.
   - `cfg.UseSyntheticMeas=0` - use real measured ranges data.
   - `cfg.UseSyntheticMeas=1` - use synthesized ranges data.
2) `cfg.scaleSigmaForBigRange` - enables scaling the measurement noise standard deviation according to the measurement range. This mechanism is implemented in the EKF and is independent of the outlier filtering mechanism.
   - `cfg.scaleSigmaForBigRange=0` - enable measurement noise scaling according to measured range.
   - `cfg.scaleSigmaForBigRange=1` - disable measurement noise scaling according to measured range.
3) `cfg.outlierFilterEnable` - enables outlier filtering mechanism with the EKF. This mechanism is independent of the sigma scaling mechanism.
   - `cfg.outlierFilterEnable=0` - enable outlier filtering mechanism with the EKF.

  - `cfg.outlierFilterEnable=1` - disable outlier filtering mechanism with the EKF.

4) `cfg.Rsp2remove` - enables to eliminate measurements corresponding to specific FTM responders in the network.
   - `cfg.outlierFilterEnable=[]` - Default. No responders are eliminated.
   - `cfg.outlierFilterEnable=[1,6]` - Example: eliminate responders 1 & 6. Notice: if an index higher than the maximal number of responders is specified, the code will ignore it.
5) additional numerical parameters that are used by the EKF implementation. May be tuned by the user if necessary.

B. Initial Estimate for the EKF States

The EKF time and position states need to be initialized before the EKF is first executed. To minimize the EKF convergence transients, the code implements a simple, "genie-like" initialization using the initial true-location of client receiver (aka "ground truth"). In practice, the information is obviously not available to the client. Alternative methods could rely for example, on range-based maximum-likelihood estimation of the client position (e.g., based on the first 3-5 ranges that appear in the measurement table), the location of the first responder to which the range was measured, (while properly setting the uncertainty regarding the position in the EKF), and so on.

C. Outlier Measurements Rejection

The code contains some basic, heuristic outlier measurements rejection mechanisms. These mechanism are implemented in the functions `RunFtmPE.m` and `KfFtmclass.m`. Additional outlier rejection mechanisms are described in e.g., [6, Chapter 15.2].

D. Performance Visualization

The code includes a function named `PlotFtmPeResults.m`, which is used to visualize the simulation results. The function outputs three figures:

1) The venue map with the location of the FTM responders, the ground-truth trajectory and the estimated client location.

2) A figure presenting various metrics of the EKF:
   - a plot of the client height state as a function of time,
   - a plot of the client combined bias state as a function of time,
   - a plot of the ratio between the client's positioning error and the EKF predicted 3-dimensional positioning error, as a function of time.
   - a plot of the CDF of the ratio between the client lateral positioning error and the EKF predicted lateral positioning error.
3) A CDF of the client positioning error.

The ratio between the client's lateral positioning error and the EKF predicted lateral positioning error is defined as follows. Let $\mathbf{p}_k = [x_k, y_k, z_k]^T$ denote the true location of the client at the kth EKF time-step, and $\hat{\mathbf{p}}_k = [\hat{x}_k, \hat{y}_k, \hat{z}_k]^T$ denote the estimated client location at the kth EKF time-step. The positioning error is defined as,

$$\epsilon_k = \|\mathbf{p}_k - \hat{\mathbf{p}}_k\| \tag{1}$$

where the notation $\|\mathbf{x}\|$ denotes a Euclidean norm. Further, let the $\mathbf{x}_k$ denote the state-vector at the kth EKF time-step. This state-vector is associated with a state covariance matrix,

$$\mathbf{P}_k = E\{(\mathbf{x}_k - \bar{\mathbf{x}}_k)(\mathbf{x}_k - \bar{\mathbf{x}}_k)^T\} \tag{2}$$

where $\bar{\mathbf{x}}_k \triangleq E\{\mathbf{x}_k\}$ and $E\{.\}$ denotes the expected value. Using the eigenvalue of $\mathbf{P}_k$ we may define the following heuristic, crude (yet simple) metric:

$$\sigma_k = \frac{\epsilon_k}{\sqrt{\sum_{i=1}^{3} \lambda_{k,i}}} \tag{3}$$

The "rule of thumb" would be to keep $\sigma_k = 1$ around 67% of the time for Gaussian-distributed measurements with range measurement noise of 1 meter. The The EKF parameters may be tuned to achieve this, using the CDF plot of σ. One should notice that if the outlier rejection mechanisms are enabled, the error is no longer Gaussian.

IV. Sample Performance

The following section provides performance examples of the reference PE when operated on the input measurements files included in the database. Fig. 1 and Fig. 2 depict the estimated & reference trajectories, along with the location of the FTM responders on the architectural blueprint of the venue. These trajectories are given for default data-set PTK1_Floor1_09.02_Location_80MHz_ch149, where Fig. 1 corresponds to the real measurements and Fig. 2 corresponds to the synthesized measurements. A graphical illustration of the output metrics of the EKF is given in Fig. 3. This

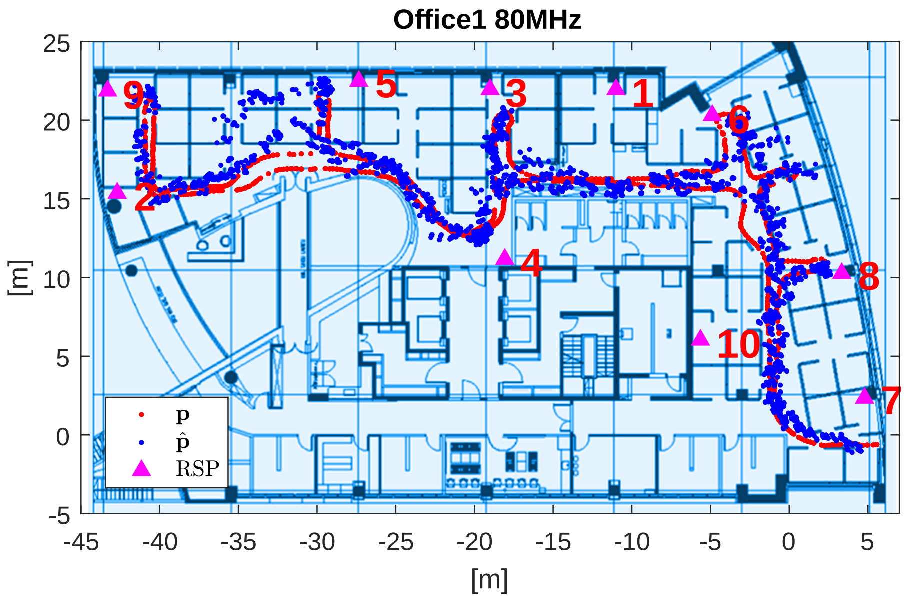

Fig. 1. Estimated & Reference Trajectory using Real Measurements

includes the tracking of the client's height estimate, a tracking of the client's range bias estimate and the a tracking of the σ_k metric over time and its CDF. The CDF in this case compares the real and synthesized data-sets. Fig. 4 compares the CDFs of the client location error (defined in (1)) for the real and synthesized measurements.

V. Summary & Conclusions

We presented a reference Matlab® implementation of a positioning engine for indoor time-delay Wi-Fi client positioning. The positioning engine is based on a linearized version of the well-known Kalman filter algorithm. The source code is provided along with a measurements database for assessing and developing time-delay based Wi-Fi client positioning systems.

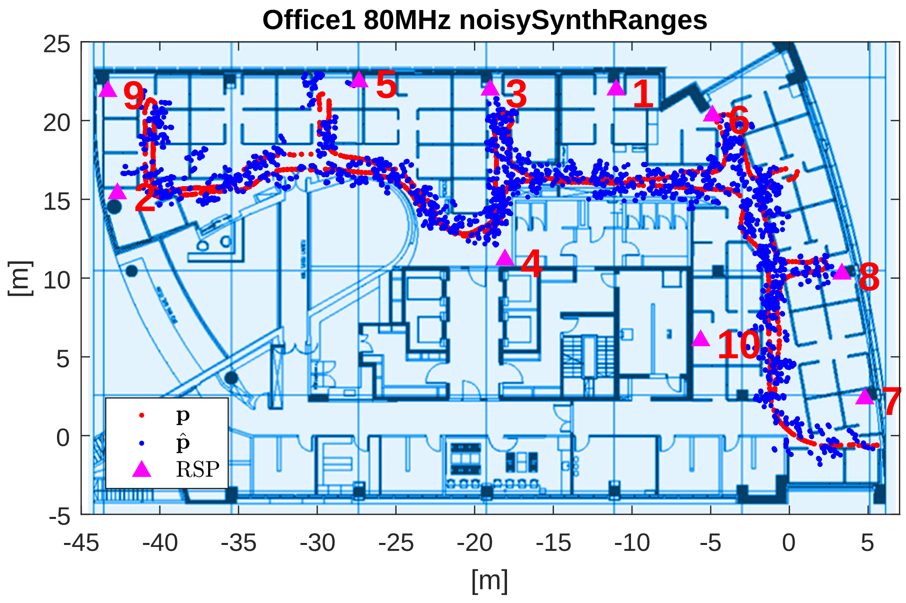

Fig. 2. Estimated & Reference Trajectory using Synthesized Measurements

References

[1] *IEEE Std 802.11^{TM}-2016 (Revision of IEEE Std 802.11-2012) - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,* IEEE 802.11 Working Group, December 7th, 2016.

[2] Leor Banin, Uri Schatzberg and Yuval Amizur, "Next Generation Indoor Positioning System Based on WiFi Time of Flight," *Proc. of The 26th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2013)* [available online: https://www.researchgate.net/publication/273443111]

[3] Uri Schatzberg, Leor Banin and Yuval Amizur, "Enhanced WiFi ToF indoor positioning system with MEMS-based INS and pedometric information," *Proc. of The 2014 IEEE/ION Position, Location and Navigation Symposium - PLANS 2014* [available online: https://www.researchgate.net/publication/269297342]

[4] Leor Banin, Uri Schatzberg and Yuval Amizur, "WiFi FTM and Map Information Fusion for Accurate Positioning," *Proc. of The 2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN), 4-7 October 2016, Alcalá de Henares, Spain* [available online: https://www.researchgate.net/publication/310057251]

[5] L. Banin, O. Bar-Shalom, N. Dvorecki and Y. Amizur, "High-Accuracy Indoor Geolocation using Collaborative Time of Arrival - Whitepaper," doc.: IEEE 802.11-17/1387R0, Sept. 2017, [available online: https://mentor.ieee.org/802.11/dcn/17/11-17-1387-00-00az-high-accuracy-indoor-geolocation-using-collaborative-time-of-arrival-ctoa-whitepaper.pdf]

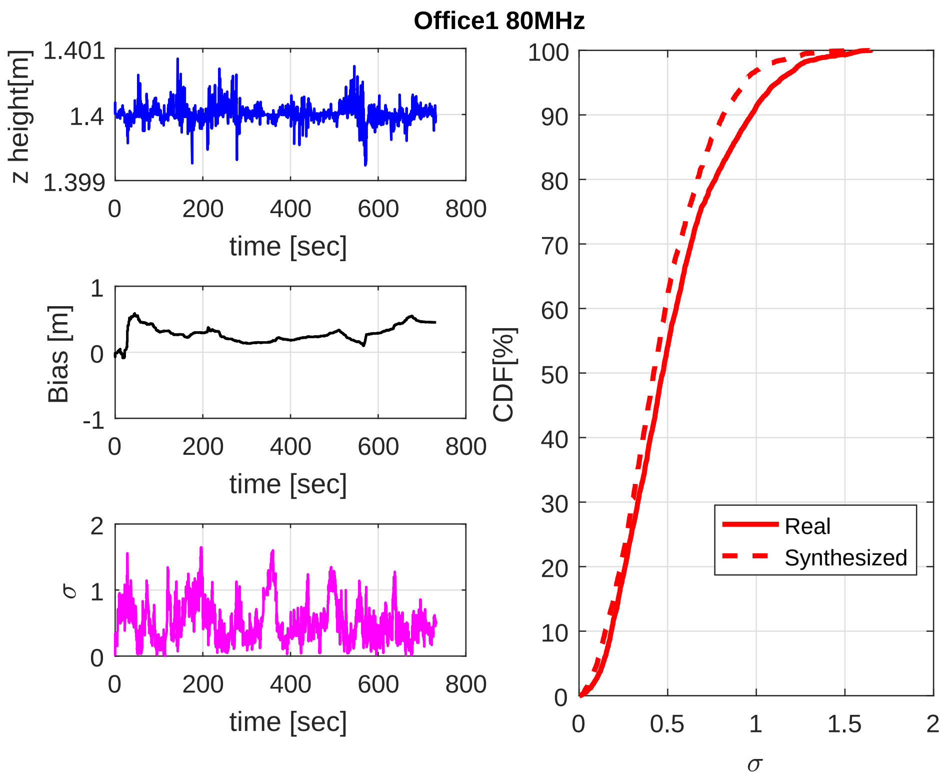

Fig. 3. EKF Metrics

[6] P. D. Groves, *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*, Artech House Boston-London, 2008.

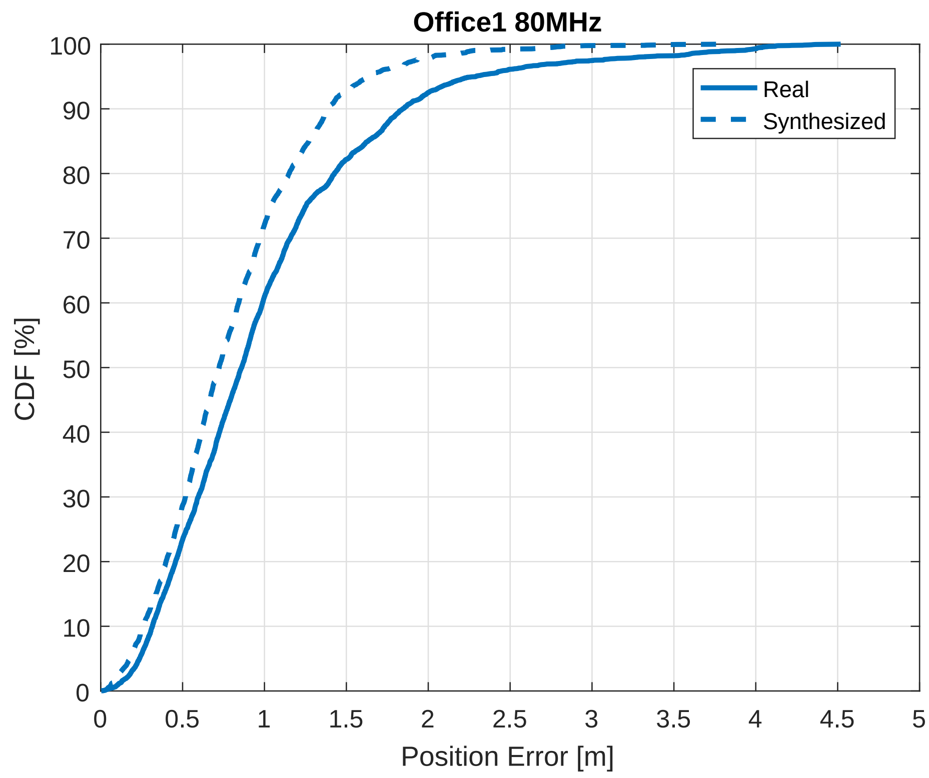

Fig. 4. Positioning Accuracy CDF