Localization of static remote devices using smartphones

D. Pedro($*$, $\dagger$), S. Tomic($\dagger$, $\ddagger$), L. Bernardo($*$, $\dagger$), M. Beko($\S$, ¶), R. Oliveira($*$, $\dagger$), R. Dinis($*$, $\dagger$), P. Pinto($*$, $\dagger$)
($*$) DEE, Faculdade de Ciências e Tecnologia, FCT, Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
($\dagger$) Instituto de Telecomunicações, Lisboa, Portugal
($\ddagger$) Institute for Systems and Robotics, IST, Lisboa, Portugal
($\S$) CTS, UNINOVA, FCT-UNL, Caparica, Portugal
(¶) CICANT-CIC.DIGITAL, Universidade Lusófona de Humanidades e Tecnologias, Lisboa, Portugal

***Abstract*—Vehicles need to locate other vehicles and network infrastructure elements on unmanned autonomous vehicle (UAV) systems. Human passengers also need to locate and be located by the vehicles, preferentially using a portable device, such as a smartphone. This paper analyses the accuracy of several localization algorithms in the remote location of entities running WiFi access points, using measurements collected in moving vehicles using a new application developed by us. The algorithms analysed include closed form estimators and one based on second order cone programming (SOCP) relaxation, which exhibits the best accuracy and is capable of estimating the path loss exponent and the transmission power. Although, due its lower complexity, the Levenberg-Marquardt algorithm was better suited for the stand-alone Android prototype application. The results show that real-time accurate positioning of static/slow moving remote entities is possible, even though the accuracy degrades when the measuring vehicle's speed increases.**[1]

***Index Terms*—Location algorithms; WiFi localization; Vehicular-based performance evaluation; Android application.**

I. Introduction

Vehicles need to locate other vehicles and network infrastructure elements on autonomous vehicle systems. Humans also need to locate and be located by the vehicles, preferentially using a portable device. This paper addresses the challenge of providing a portable localization system that maps static or slow moving remote nodes without requiring any known anchor or map, using only the smartphone's sensors, Android's services, and the energy received from the remote nodes. The idea is not to locate the terminals location, but the location of a remote node sending a given SSID.

Fingerprinting and multilateration are two main approaches for WiFi positioning [1]. Fingerprinting-based WiFi positioning matches the measured Received Signal Strength (RSS) values to a pre-surveyed radio map database [2]. Multilateration-based WiFi positioning calculates the ranges between the device and Access Points (APs) using a wireless signal propagation model; trilateration or multilateration algorithms, such as the ones in [3], [4], are used to estimate the position of the nodes. Fingerprinting usually provides more accurate position solutions for a terminal location [1], but fails to provide the exact location of a remote node. Multilateration is the best approach to locate remote nodes, requiring only the relative position of nodes [5].

Localization precision can be increased by using enhanced GNSS services (e.g. Assisted-GPS (A-GPS) [6]), Angle of Arrival (AoA) (e.g. [7]) and the micro-electro-mechanical systems (MEMS) sensors present in the portable devices (e.g. [1]). MEMS sensors can be used to implement inertial navigation system (INS) and/or pedestrian dead reckoning (PDR) for vehicle or pedestrian navigation applications.

Simultaneous Localization and Mapping (SLAM) algorithms address the localization problem assuming no prior site information. Most SLAM solutions combine PDR/INS and fingerprinting, although with more elaborate robotic sensors and mobility scenarios [8]. SLAM is a nonconvex problem and most SLAM algorithms are based on iterative nonlinear optimization. State-of-the-art iterative solvers fail to converge to a global minimum of the cost function for relatively small noise levels, so methods that explored the nonconvexity nature in SLAM and alternative maximum likelihood formulations were proposed, that allow a global optimization solutions using (convex) semidefinitive programming (SDP) [8]. Other authors proposed equal efficient and less complex convex relaxation solutions [9]. Although different, evolution of multilateration algorithms followed a similar path, and some of the best performing ones are also based on convex relaxation solutions (e.g. [4], [10]).

This paper proposes a multilateration approach for remote node localization. Several multilateration algorithms were analysed using the same RSS samples collected by a smartphone in a vehicular scenario.The emphasis in this paper is identifying the multilateration algorithms that best handle the input measurement errors. The paper's main contributions is the analysis and performance evaluation of multilateration techniques, comparing classical static ones, maximum likelihood formulations, convex SDP and convex relaxation solutions. A new Android application that supports remote localization of APs is also presented.

This paper is organized as follows: the system overview is presented in section II. Multilateration algorithms are reviewed in section III, and their performance is evaluated in section IV. Section V summarizes the main conclusions.

[1] This work was funded by FCT/MEC through national funds and when applicable co-funded by FEDER – PT2020 partnership agreement under the project UID/EEA/50008/2013 and Program Investigador FCT under Grant IF/00325/2015.

II. System architecture

The system is designed for a mobile terminal to track the location of static or slow moving entities running WiFi APs. The approach followed is illustrated in figure 1. A moving terminal collects in multiple locations RSS measurements of the signal received from the APs, recording the terminal location at each measurement. This paper focuses exclusively in a non-cooperative implementation, where the multilateration algorithm uses only the terminal's RSS vectors. But a combination of vectors from several terminals could be used.

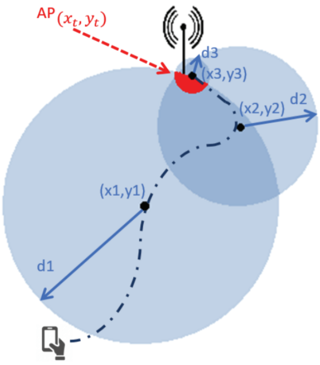

Fig. 1. Localization scheme.

The main component is the software running in the terminal, which implements the modules depicted in figure 2. The localization module is capable of switching between an outdoor mode and an indoor mode (not described in this paper). In outdoor mode, it uses the Android localization service, which combines GNSS, WiFi and cellular signals to estimate the location. An odometry module is used in indoor mode, to track the relative position of the terminal from the last known global position. A WiFi scanner module is run concurrently, which collects the RSS vectors using the *android.net.wifi* Android's library. In the non-cooperative mode, the vectors are fed to the multilateration algorithm, implemented at the FindAP module, which returns the location of the selected AP. A prototype Android application was implemented[2] for the non-cooperative mode. The current prototype application provides two visualization mechanisms for the AP's location: it can provide an arrow pointing to the direction towards the AP (fig. 3); or, it can provide the location of the APs in a map (fig. 4). It was developed mainly for human users. However, it can be easily extended to an UAV scenario: the FindAP module can send the APs' locations directly to an autonomous driving unit inside the vehicle and the localization module could receive the location from the vehicle's odometry unit (that may use sensors attached to the wheels).

For testing purposes, the application was extended to collect AP's RSS measurements, time, odometry/android location, allowing the evaluation of multiple algorithms with real measured data, which is presented in this paper.

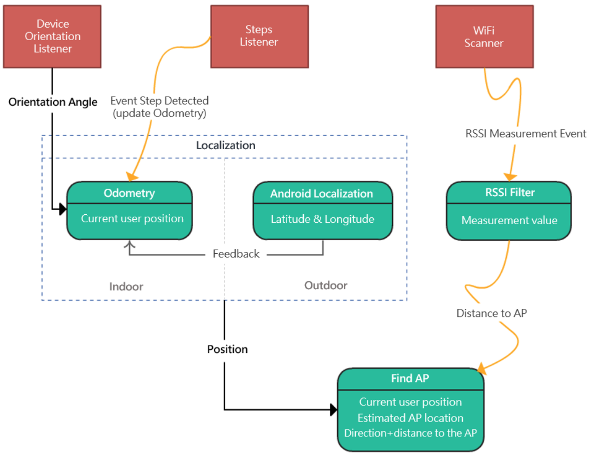

Fig. 2. Terminal operation scheme.

Fig. 3. Find AP interface

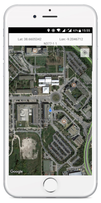

Fig. 4. Map interface

III. Localization algorithms

The problem of localizing a remote node is dual to the classical localization problem. Let t denote the target with unknown coordinates and $s_i, i = 1, ..., N$, the set of known positions where the terminal collected the target's RSS values, P_i^r. The distance corresponding to a given RSS value P^r can be obtained using the Friis equation,

$$d = 10^{\frac{\alpha - P^r - 10\gamma \log_{10}(f)}{10\gamma}}, \tag{1}$$

where γ represents the path loss exponent (PLE), f the carrier frequency and α the contributions of the antennas gains and the transmission power (assumed uniform) [11]. The measured distance is defined by

$$d_i = \|s_i - t\| + \varepsilon_i, \tag{2}$$

where ε_i is the error in the ith measurement, which among others, includes contributions from the terminal location error and due to errors in the estimated PLE value. Errors are assumed independent. Equation (2) can be generalized to $D = h(t) + \varepsilon$, where D denotes the measurement vector, h the vector-valued measurement function and ε the error vector. The objective of a positioning method is to find the position t that minimizes the residual between the true distances and the measured ones, which defines a multilateration problem. One way for obtaining an estimate of the target's location, t^*, is via the least square (LS) criterion, i.e,

$$t^* = \arg\min_t \|D - h(t)\|^2 = \arg\min_t \sum_{i=1}^{N} (d_i - h_i(t))^2. \tag{3}$$

[2] The app source code can be downloaded from https://github.com/dario-pedro/wifi_finder.

In this paper we compare a set of algorithms that might be suitable for our application due to their high accuracy and low complexity. We analyse the performance of the following well known closed form estimators for overdetermined systems:

- Simple - simple intersection method is derived by expanding and solving the squared range equations to $2s_i^T t = \|t\|^2 + \|s_i\|^2 - d_i^2$ [3];
- Range-Bancroft - this method is also based on an expansion of the squared range equations presented above, but it uses Moore-Penrose matrices' pseudo-inverses to obtain estimate solutions and select the one with minimal residual value [3];
- Beck - Beck *et al.* [12] defined a procedure for computing the exact least-quartic solution for the range equations. The procedure uses the bisection method to find a root of a univariate strictly monotonous function on an interval that is easily computed;
- Cheung - Cheung *et al.* [13] gave a constrained weighted least squares solution for range measurements. It assumes each measurement error ε_i is a zero-mean white Gaussian process with known variance σ_i^2.

Additionally, we consider two closed form iterative algorithms and review an estimator based on a convex relaxation solution [4], presented in the subsections below.

A. Gauss-Newton

Gauss-Newton method is an iterative method to minimize a sum of squared function values for scenarios with additive noise with finite variance. In this paper we consider the regularized Gauss-Newton method [3], which reduces the divergence problems using an algorithm that is equivalent to a Bayesian maximum-a-posteriori algorithm with prior distribution chosen around the center point of the terminal positions. Let t_d denote the mean point of $s_i, i = 1, ..., N$ and c a regularisation coefficient. The results presented in section IV were obtained by using a maximum number of iterations of $K_{max} = 8$, $c = 10^{-4}$ and the stopping tolerance $\delta = 2 * 10^{-2}$.

B. Levenberg-Marquardt

The Levenberg-Marquardt (LM) algorithm is an alternative iterative technique that locates the minimum of a function expressed as the sum of squares of nonlinear functions [14]. LM can be thought of as a combination of gradient descent and the Gauss-Newton method. When the current solution is far from the correct one, the algorithm behaves like a gradient descent method: slow, but guaranteed to converge. When the current solution is close to the correct solution, it becomes a Gauss-Newton method. This is implemented using a dampening factor ϑ, which is calculated in each iteration. A weight $w_i, i = 1, ..., N$ was used in the sum, with a value $w_i = 1/{d_i}^2$. To avoid divergence problems due to matrices over-dimensioning, the measured distances were bound to $d_i = \max(d_i, 10^{-7})$.

The algorithm starts by calculating the initial starting position, $t_0 = t_d$ and defining the stopping tolerance δ (10^{-4} times the function tolerance). In each step, the Jacobian J is calculated, and $t_{k+1} = t_k + \Delta t_k$ where Δt_k is the least-squares solution to $(J^T J + \vartheta \mathrm{diag}(J^T J))\Delta t = J^T \varepsilon$. The dampening factor ϑ is updated based on the success of the last t_k update. The cycle stops when $\|\Delta t_k\| < \delta$, or when K_{max} is reached.

LM algorithm was initially implemented using the Matlab library, and latter selected for the non-collaborative Android prototype, where it was implemented using the *org.apache.commons.math3.fitting.leastsquares.Levenberg MarquardtOptimizer* library, with a maximum number of iterations $K_{max} = 1000$.

C. Tomic SOCP Relaxation

Tomic *et al.* [4] derived a convex estimator, by tightly approximating the ML (Maximum Likelihood) estimator for small noise. This estimator is based on SOCP relaxation technique. Contrarily to the previous algorithms, this algorithm handles γ and the transmission power P_T as unknowns, estimating them simultaneously with the t's position. The following path loss model (dB) was considered,

$$L_i = L_0 + 10\gamma \log_{10} \|t - s_i\|/d_0 + v_i, \; i = 1, \ldots, N, \quad (4)$$

where L_0 denotes the value of losses along the path at a short reference distance d_0 ($\|t - s_i\| \geq d_0$) and v_i is the log-normal shadowing term modeled as a zero-mean Gaussian random variable with variance σ_i^2, i.e. $v_i \sim \mathcal{N}\left(0, \sigma_i^2\right)$. This formulation combines (1) and (2) in a single equation.

To formulate the joint localization problem, the ML criterion was considered, but finding the ML estimator, $\widehat{\theta}$, implies solving the non-linear and non-convex least-squares problem,

$$\widehat{\theta} = \underset{\theta=[t;L_0;\gamma]}{\arg\min} \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \times \left[\left(L_i - h^T\theta\right) - 10 g^T \theta \log_{10} \frac{\|C^T\theta - s_i\|}{d_0}\right]^2, \quad (5)$$

where $h = [0_{2\times1}; 1; 0]$, $g = [0_{3\times1}; 1]$ and $C = [I_2; 0_{2\times2}]$. The problem defined by (5) is not convex and has no closed-form solution. Therefore, the solution proposed in [4] estimates the position of interest following the iterative procedure:

1) Set the initial estimate of $\gamma, \widehat{\gamma^0} \in [\gamma_{min}, \gamma_{max}]$, and the iterator counter $k = 1$.
2) Solve the SOCP problem [4]: $\underset{t,g,z,\eta,p}{\text{minimize}}\; p$

$$\text{subject to } \left\|\left[2z \quad p - s_i\right]\right\| \leq p + 1, \|t - s_i\| \leq g_i, \\ z_i = \psi_i g_i - \eta d_0, \; i = 1, \ldots, N. \quad (6)$$

3) Use $\widehat{\gamma}^{k-1}$ and $\widehat{t}^{k-1}$ to compute L_0's ML estimative,

$$\widehat{L}_0^k = \sum_{i=1}^{N} \left(L_i - 10\widehat{\gamma}^{k-1} \log_{10} \frac{\left\|\widehat{t}^{k-1} - s_i\right\|}{d_0}\right)/N \quad (7)$$

4) Use $\widehat{t}^{k-1}$ and $\widehat{L}_0^k$ to find the ML estimative of γ, $\widehat{\gamma}$

$$\widehat{\gamma}^k = \frac{\sum_{i=1}^{N} 10 \log_{10} \frac{\left\|\widehat{t}^{k-1} - s_i\right\|}{d_0} \left(L_i - \widehat{L}_0^k\right)}{\sum_{i=1}^{N} \left(10 \log_{10} \frac{\left\|\widehat{t}^{k-1} - s_i\right\|}{d_0} L_i - \widehat{L}_0^k\right)^2} \quad (8)$$

If $\widehat{\gamma}^k \notin [\gamma_{min}, \ \gamma_{max}]$ the process should be stopped at this point and use $\widehat{t}^{k-1}$ as the final estimate.

5) If $k > K^T_{max}$ (K^T_{max} represents the maximum number of iterations) it must stop and consider $\widehat{t}^k$ as estimate ; Otherwise repeat step 2 with the help of $\widehat{\gamma}^k$ and $\widehat{L}^k_0$ and increment k.

IV. Performance evaluation

The performance of the localization algorithms was analysed for a smartphone inside a moving vehicle using the location information provided by A-GPS to the Android service, which has an expected accuracy for a static terminal of around 8 m [15]. A static AP was placed in the location represented in figure 5 with $P_T = 28$ dBm. The RSS measurements were registered every second using a OnePlus 2 smartphone running Android 6.0.1. The application allows the user to export the data, which was later swot in MATLAB, and all algorithms described in section III were implemented and run using the same data. Moreover, $\gamma^0 = 2.5$, $\gamma_{min} = 1.5$, $\gamma_{max} = 3.5$ and $K^T_{max} = 3$ were considered for the algorithms that require these parameters. The metric used to compare the accuracy of one estimated location was the Euclidean distance to the exact location of the test AP. The overall performance metric considered for each algorithm was the Root Mean Square Error (RMSE), $\sqrt{\sum_{i=1}^{M} \left\| t - \widehat{t}_i \right\|^2 / M}$, where M denotes the size of the measurement set.

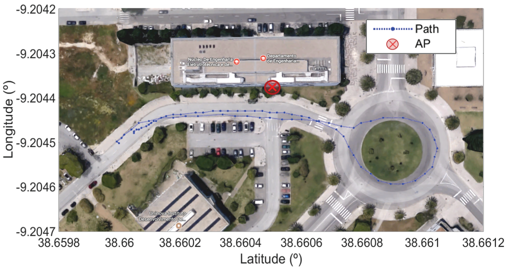

Fig. 5. Vehicular testing scenario.

Two sets of tests were done following the path depicted in figure 5, inside Caparica Campus of NOVA University of Lisbon, at two different vehicle average speeds: 20 km/h and 30 km/h. Figure 6 depicts the RSS values measured in two of the tests. Due to the constant sampling rate, a lower number of samples was measured for the higher speed and the measured RSS values have a narrower variation range, probably due to some filtering implemented at the Android library. RSS values stay low during the first measurements, and start to increase until a first maximum, when the car passed in front of the AP coming from the left in the right lane in figure 5. RSS decreases and has a second maximum value when the car drives to and goes around the roundabout and passes a second time in from of the AP coming from the right in the opposite lane, before returning to the starting point.

Figures 7 and 8 depict the distance between the estimated and the exact location of the test AP, after the ith measurement for the two speeds. Note that the ith sample uses all measurements of the previous samples, and some variations may occur due to measurements with higher error being added.

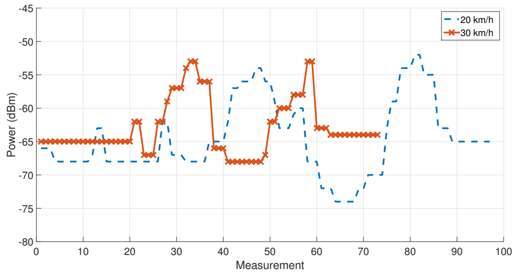

Fig. 6. RSS measured at 20 km/h and 30 km/h.

It can be seen at figures 7 and 8 that the accuracy decreased for the highest speed, most likely due to the less accurate RSS and vehicle location values. At 20km/h it was possible to have initial location estimations (after 2 or 3 RSS samples) with an accuracy below 40m, whereas for 30km/h it was only possible to have an accuracy below 50m for two of the methods. Nevertheless, these measurements require more RSS samples to stabilize, as anticipated. The accuracy remains stable until the first significant RSS variation occurs, i.e, when the vehicle approaches the AP and goes around the roundabout. The overall accuracy is presented in table I for two regions, RMSE1 and RMSE2. RMSE1 corresponds to the samples between 1 and 50 for 20km/h and between 1 and 38 for 30km/h; the "breaking point" was the position at which the vehicle reaches the farthest point from the AP in the roundabout. RMSE2 contains the remaining samples, but the localization uses all the previous samples, including RMSE1 ones. Therefore, it includes more samples with higher RSS values, that were more successfully used by some of the algorithms to increase the accuracy. It can be seen that Simple and Gauss-Newton have the worst RMSE2 accuracy, while Tomic SOCP algorithm tends to have the best accuracy, as expected. This is probably due to considering more unknowns during the optimization, and its increased computational complexity that was able to compensate part of the vehicle location error, achieving an accuracy of 4.18 m, far below the A-GPS accuracy at 20km/h. On the other hand, Levenberg-Marquardt algorithm performance was the best within the closed form estimators, with an accuracy of 6.58 m, also below A-GPS' accuracy. Its accuracy beat Tomic SOCP's for RMSE2 in the 30km/h scenario, where the RSS and vehicle location errors were much higher. It is worth mentioning that the closed-form estimators were run with predefined γ and P_T values, which introduce additional errors, since the γ tends to vary during the course. Thus, they are less prepared for differentiated and/or cooperative scenarios, where P_T and γ values considered might not be optimal.

Table I also presents the average calculation time used by the algorithms in MATLAB with a RSS vector with 30 samples. In order to run the application in real-time, mitigate old

TABLE I
AVERAGE ERROR

Algorithm	Runtime (ms)	20 km/h		30 km/h	
		RMSE1 (m)	RMSE2 (m)	RMSE1 (m)	RMSE2 (m)
Simple Intersection	8.66	47.76	77.16	65.35	77.64
Bancroft	12.31	40.61	14.95	44.83	17.72
Gauss-Newton	26.5	36.51	43.42	52.53	44.33
Beck	26.8	36.25	10.5	47.56	14.7
Cheung	15.2	36.26	10.48	47.46	14.68
Levenberg-Marquardt	249	39.44	6.58	60.51	12.47
Tomic SOCP	$3{,}91\times10^4$	44.83	4.18	53.24	14.26

readings effects and process estimations without influencing the application fluidity, we decided to limit the RSS measurements stored in the application prototype to 30. It can be seen that Levenberg-Marquardt is the second more computationally intensive algorithm in the Matlab implementation, but it is about 100 times lighter than Tomic SOCP. Additionally, using the *org.apache.commons.math3* library and considering the maximum vector size of 30 samples, we were able to measure an average run time of 13.47 ms with the prototype application running in the smartphone. Therefore, due to its excellent trade-off between accuracy and execution time, this algorithm was chosen for android application implementation for non-cooperative localization. However, we plan to use a convex-based estimator, like the Tomic SOCP, for the cooperative localization implementation, due to its higher accuracy.

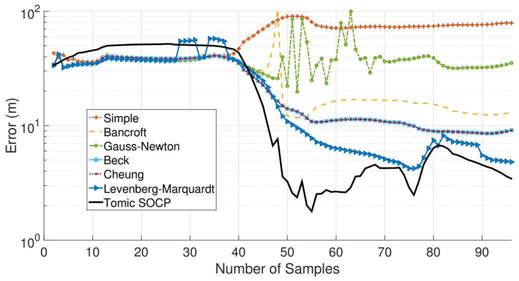

Fig. 7. Location accuracy at 20 km/h.

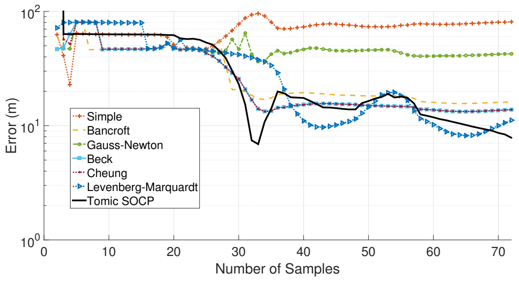

Fig. 8. Location accuracy at 30 km/h.

V. Conclusion

This paper compared the performance of several localization algorithms, using real RSS data measurements acquired through a smartphone inside a moving vehicle, considering a stand-alone operation. It also presented the prototype Android application that implements the best suited algorithm, which for the considered non-cooperative scenario was the LM algorithm. Our experimental results showed that it is possible for a vehicle to estimate the position of an AP node using only 3 samples with an error of 40 m. But the error can be reduced by having more samples. Therefore, these can be obtained by regulating the sample period according to the vehicles speed. Although, further studies are needed to understand what is the maximum vehicle speed that is effectively supported by the application, due to the higher RSS and vehicle location errors for higher speeds. This paper only studied the GNSS based smartphone' location measurements. Odometry based measurements will be covered in a future paper. Future work will include the study of the performance of the localization algorithms in a cooperative scenario, where the additional estimation capabilities of the Tomic SOCP and new algorithms using relaxation methods are expected to contribute more in a cooperative context.

References

[1] Y. Zhuang, H. Lan, Y. Li, and N. El-Sheimy, “PDR/INS/WiFi integration based on handheld devices for indoor pedestrian navigation,” *Micromachines*, vol. 6, no. 6, pp. 793–812, 2015.

[2] P. Bahl and V. Padmanabhan, “RADAR: an in-building RF-based user location and tracking system,” in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, vol. 2. IEEE, pp. 775–784.

[3] N. Sirola, “Closed-form algorithms in mobile positioning: Myths and misconceptions,” in *Proc. 2010 7th Work. Positioning, Navig. Commun. WPNC'10*. IEEE, mar 2010, pp. 38–44.

[4] S. Tomic, M. Beko, and R. Dinis, “RSS-based localization in wireless sensor networks using convex relaxation: Noncooperative and cooperative schemes,” *IEEE Trans. Veh. Technol.*, vol. 64, no. 5, pp. 2037–2050, may 2015.

[5] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan, “Indoor localization without the pain,” in *Proc. Sixt. Annu. Int. Conf. Mob. Comput. Netw. - MobiCom '10*. New York, New York, USA: ACM Press, 2010, p. 173.

[6] P. A. Zandbergen, “Accuracy of iPhone locations: A comparison of assisted GPS, WiFi and cellular positioning,” *Trans. GIS*, vol. 13, no. SUPPL. 1, pp. 5–25, jun 2009.

[7] S. Tomic, M. Beko, and R. Dinis, “Distributed RSS-AoA Based Localization With Unknown Transmit Powers,” *IEEE Wireless Communications Letters*, vol. 5, no. 4, pp. 392–395, Aug 2016.

[8] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, “Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age,” *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, dec 2016.

[9] M. Liu, S. Huang, G. Dissanayake, and H. Wang, “A convex optimization based approach for pose SLAM problems,” in *IEEE Int. Conf. Intell. Robot. Syst.* IEEE, oct 2012, pp. 1898–1903.

[10] S. Tomic, M. Beko, R. Dinis, and P. Montezuma, “A Closed-Form Solution for RSS/AoA Target Localization by Spherical Coordinates Conversion,” *IEEE Wireless Communications Letters*, vol. 5, no. 6, pp. 680–683, Dec 2016.

[11] T. S. Rappaport, *Wireless Communications: Principles and Practice, 2nd edition*. Prentice Hall, Jan 2002.

[12] A. Beck, P. Stoica, and Jian Li, “Exact and Approximate Solutions of Source Localization Problems,” *IEEE Trans. Signal Process.*, vol. 56, no. 5, pp. 1770–1778, may 2008.

[13] K. W. Cheung, H. C. So, W. K. Ma, and Y. T. Chan, “Least Squares Algorithms for Time-of-Arrival-Based Mobile Location,” *IEEE Trans. Signal Process.*, vol. 52, no. 4, pp. 1121–1128, apr 2004.

[14] H. P. Gavin, “The levenberg-marquardt method for nonlinear least squares curve-fitting problems,” Dept. Civil Environ. Eng., Duke Univ., Durham, NC, USA, Tech. Rep., Oct. 2013.

[15] P. A. Zandbergen, “Accuracy of iPhone locations: A comparison of assisted GPS, WiFi and cellular positioning,” in *Trans. GIS*, vol. 13, no. SUPPL. 1. Blackwell Publishing Ltd, jun 2009, pp. 5–25.