IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 30, 2018, accepted March 9, 2018, date of publication March 21, 2018, date of current version April 23, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2817800

Improving Indoor Localization Using Convolutional Neural Networks on Computationally Restricted Devices

KLEMEN BREGAR AND MIHAEL MOHORČIČ, (Senior Member, IEEE)
[1]Department of Communication Systems, Jožef Stefan Institute, 1000 Ljubljana, Slovenia
[2]Jožef Stefan International Postgraduate School, 1000 Ljubljana, Slovenia

Corresponding author: Klemen Bregar (klemen.bregar@ijs.si)

This work was supported in part by the Slovenian Research Agency Through the Young Researcher Scheme under Grant P2-0016 and Grant L2-7664 and in part by the European Community through the H2020 eWINE Project under Grant 688 116.

ABSTRACT Indoor localization is one of the key enablers for various application and service areas that rely on precise locations of people, goods, and assets, ranging from home automation and assisted living to increased automation of production and logistic processes and wireless network optimization. Existing solutions provide various levels of precision, which also depends on the complexity of the indoor radio environment. In this paper, we propose two methods for reducing the localization error in indoor non-line-of-sight (NLoS) conditions using raw channel impulse response (CIR) information obtained from ultra-wide band radios requiring no prior knowledge about the radio environment. The methods are based on NLoS channel classification and ranging error regression models, both using convolutional neural networks (CNNs) and implemented in the TensorFlow computational framework. We first show that NLoS channel classification using raw CIR data outperforms existing approaches that are based on derived input signal features. We further demonstrate that the predicted NLoS channel state and predicted ranging error information, used in combination with least squares (LS) and weighted LS location estimation algorithms, significantly improve indoor localization performance. We also evaluate the computational performance and suitability of the proposed CNN-based algorithms on various computing platforms with a wide range of different capabilities and show that in a distributed localization system, they can also be used on computationally restricted devices.

INDEX TERMS Channel impulse response, convolutional neural network, deep learning, indoor localization, non-line-of-sight, ranging error mitigation, ultra-wide band.

I. INTRODUCTION

Information about indoor location is increasing in importance in modern communication services and applications. This information can be used to extend tracking and navigation services for people and goods from outdoor to indoor environments, for guiding autonomous vehicles inside manufacturing facilities, and for wireless network optimization to facilitate the above applications.

Next-generation wireless communication systems will need to serve an increasing number of mobile users, numerous connected Internet-of-Things (IoT) devices and smart vehicles with greater bandwidths and shorter response times. The required increase in capacity of mobile wireless networks can be obtained with a combination of macrocells and small cells connected in self-organizing, heterogeneous, dynamic wireless networks. In the most demanding environments, where a large number of devices need high quality of service (QoS), dense small-cell networks will be required.

In a dynamic dense small cell network, where many users are constantly moving, many handoff events occur on a regular basis. For desired QoS provisioning and thus preventing overloading individual cells, smart and efficient handoff algorithms are needed. Precisely tracked user's location can be used for predicting the user's future location and thus forecasting upcoming handoff events. With sufficient context information, a system controller can reserve resources for new users and predict their release when the user leaves the cell coverage area.

Global navigation satellite systems (GNSS) such as global positioning system (GPS) are generally sufficient for outdoor

applications because of their accuracy and broad availability in smartphones and other devices. For use in indoor environments, however, their signals are too weak or degraded by multipath effects, requiring different approaches. Indoor localization systems require special infrastructure, which can be part of existing communication means (e.g., WiFi access points) or needs to be installed additionally (e.g., Bluetooth beacons). These systems are based on signal propagation characteristics, and for greater localization performance, they mostly need extensive, time-consuming measurements and calibration procedures.

An alternative approach, enabled by ultra-wide band (UWB) radios, is based on measuring the time-of-flight (ToF) between two nodes and enhanced by detailed information about propagation channel characteristics obtained from channel impulse response (CIR). Propagation characteristics derived from CIR information can be used to mitigate ToF deviations caused by prolonged signal propagation paths because of obstacles in the environment, thereby significantly improving localization performance.

In this article, we propose a novel approach to indoor localization using two convolutional neural network (CNN) algorithms working on raw CIR data as obtained from UWB radio for each received packet. The first algorithm, used for detecting non-line-of-sight (NLoS) channel conditions, eliminates the need for feature extraction procedures. The second algorithm for ranging error regression is used to reduce the impact of NLoS range estimation on localization accuracy by weighting individual range contributions according to the estimated ranging error. To investigate the suitability of complex CNN channel classification and ranging error regression models on computationally restricted devices on the edge of the network, we also conducted performance comparisons in terms of calculation times on a range of various computing platforms.

The main contributions of this paper are as follows:

- A new approach to indoor localization using CNN for channel classification and ranging error regression on 1-dimensional raw CIR traces rather than on derived features.
- Performance evaluation of proposed classification algorithm on different computing platforms, ranging from low-cost embedded computers to high-performance personal PCs.

The remainder of this article is organized as follows. Section II summarizes some related work from the literature. Section III outlines the localization system architecture, and Section IV describes the measurement equipment and the process of constructing NLoS classification and localization datasets. In Section V, two node localization methods are described that are subsequently used in combination with the newly proposed CNN-based channel classification and localization error mitigation algorithms defined in Section VI. In Section VII, different localization algorithms are tested using the proposed NLoS detection and ranging error models based on CNN, and in Section VIII, a critical performance evaluation of the proposed channel classification system is conducted on several computing platforms with different computational capabilities. Section IX concludes the paper.

II. RELATED WORK

Different indoor localization approaches in the literature can be coarsely classified into three main groups according to the modeling information that they are based on: received signal strength indicator (RSSI)-based methods, angle-of-arrival (AoA)-based methods and time-based localization algorithms [1].

The most prominent source of errors in RSSI-based localization algorithms comes from the high dependence on channel variability caused by the dynamic and unpredictable nature of radio channel effects (shadowing, multipath propagation, reflections, channel fading, and so forth) [2], [3]. Most of the advanced and precise RSSI-based indoor localization methods are RSSI fingerprinting methods [3], [4], where a device is positioned based on the RSSI fingerprint captured from several access points (APs). However, most of those methods are complex and require extensive environment measurements and calibration. Adaptive fingerprinting algorithms were recently developed to eliminate the need for periodical complete system recalibration because of the dynamic environment nature [5].

AoA-based localization systems rely on the captured direction of propagation of the radio wave at the receiving antenna array. If great angle resolution and consequently great localization accuracy are desired, then complex antenna arrays are needed on the receiver side [6], which is not practical or even not feasible in most cases.

Time-based localization algorithms are based on measuring the signal propagation time between a transmitter and a receiver. Time-of-arrival (ToA) or time-difference-of-arrival (TDoA) localization algorithms can be implemented if precise clock synchronization is ensured among all or reference nodes inside the system, respectively [1].

In indoor positioning of resource-constrained wireless devices, such as that considered in this study, there is generally no clock synchronization available between the nodes. In such a case, two-way ranging (TWR) or round-trip time (RTT) methods can be used. In TWR ranging systems, every packet is timestamped on both sides and travels two times between the transmitter and the receiver. With this approach, two timestamps on a receiver and two timestamps on a transmitter are collected. This eliminates the local clock differences in time-of-flight (ToF) calculation, but it requires good clock stability to reduce the influence of local clock variability and accurate and efficient start of frame timestamp determination [7] since each nanosecond of error in ToF means approximately 30 cm of error in range estimation. UWB pulse radios with their ultra-wide bandwidth (typically more than 500 MHz) and very short transmit pulses offer high temporal and spatial resolutions and great multipath fading immunity compared to narrow-band carrier-based communication technologies. However, great multipath resolvability

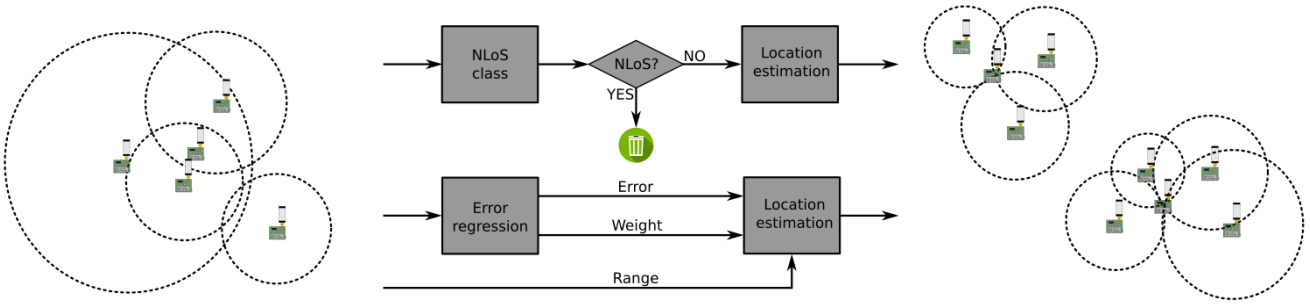

FIGURE 1. The figure shows two approaches, how ranging error and its contribution to the localization accuracy can be mitigated. The first approach detects NLoS condition and removes NLoS ranges from localization process. The second approach estimates the ranging error and uses it as a weight in the localization process.

alone does not eliminate the effects of multipath and NLoS propagation [8], [9].

Ranging errors introduced by multipath and NLoS propagation in indoor environments can easily achieve ranges of meters and must be properly detected and mitigated to prevent larger localization errors. Localization error mitigation approaches can be classified as NLoS identification techniques and range error mitigation techniques. NLoS identification is used to properly detect NLoS nodes, which can later be eliminated from the pool of nodes used for localization [10]. This is useful when we have a large number of anchor nodes available, many of them with an LoS link to a localized node. NLoS identification techniques mostly use channel and waveform statistics (RSSI, kurtosis, skewness, mean excess delay, and so forth) as input data. Some of them are based on likelihood ratio tests or binary hypothesis tests [9], [11], which rely on probability distribution functions of various parameters for current channel realization. Other methods are based on machine learning algorithms, such as support vector machine (SVM) [10], [12], Gaussian processes (GP) [11], and relevance vector machine (RVM) [12].

Range error mitigation is performed in a similar way as NLoS identification. Some works use binary hypothesis testing with propagation and error models [11], but most examples in the literature are based on the same machine learning algorithms as NLoS identification. In many cases, authors predict the ranging error based on channel characteristics and subtract it from previously estimated range [10], [12], [13], but in other cases, they estimate the weights used for the additional weighting of mitigated ranges to further improve localization accuracy [12].

Recently, training algorithms for neural networks were improved to an extent where fully dimensional signals such as images, waveforms, and so forth can be fed to a classification algorithm without any complex feature extractions and input transformations. With the advent of affordable general-purpose graphics processing units (GPGPUs) and improvements in deep learning neural network training mechanisms, CNNs as their subset have become increasingly popular and feasible for production-ready use. Architectural ideas for CNNs (e.g., receptive fields, shared weights, spatial subsampling, and so on) are drawn from their biological counterparts and applied to ensure some degree of input shift, scale and distortion invariance [14]. To the best of our knowledge, there has been no prior implementation of NLoS classification or ranging error mitigation based on raw CIR data using CNNs.

III. SYSTEM ARCHITECTURE

The architectures for two indoor localization systems, one based on NLoS range classification and the other based on ranging error estimation, are conceptually depicted in Fig. 1 along with graphical representations of range manipulation in both approaches. The upper system uses NLoS classification with a procedure to eliminate all NLoS ranges from the localization anchor pool. It consists of several functional parts. The input classification block with a NLoS classification unit recognizes NLoS range measurements based on CIR data accessible for each range measurement. If a range measurement for a given pair of nodes is recognized as a NLoS measurement, then the corresponding node is removed from the pool of available anchor nodes for localization; otherwise, it is fed directly to the location estimation unit.

The second localization system depicted in the lower part of Fig. 1 uses a CNN for ranging error estimation and subsequent reduction of localization error without reducing the number of available localization anchor nodes by weighting individual range contributions according to the estimated ranging error. The input error regression block estimates the ranging error size based on a model constructed during the training of the neural network with localization dataset described in Sect. IV-C. The ranging error estimates are then used in the localization processes for removing the predicted error from the estimated ranges, or in the case of the weighted-least-squares-based (WLS) location estimator, also for weighting individual range estimates.

IV. MEASUREMENTS

This section outlines the procedures and measurement equipment used for collecting the indoor UWB NLoS classification dataset and localization error mitigation dataset.

The measurement campaign was split into two phases. The first phase included the collection of LoS and NLoS data in various representative indoor environments without range measurements. The second phase included measurements in an office environment with nodes positioned on a predefined grid with known exact distances between nodes.

A. MEASUREMENT EQUIPMENT AND SETUP

For experimentation with different localization approaches, we developed a custom standalone radio board, which is depicted in Fig. 2. It is based on an impulse radio UWB (IR-UWB) technology using a currently available low-cost transceiver, DecaWave DWM1000[1]. DWM1000 is an IEEE 802.15.4-2011 IR-UWB compliant wireless transceiver module with an integrated ceramic antenna that allows indoor ranging with a precision of 10 cm.

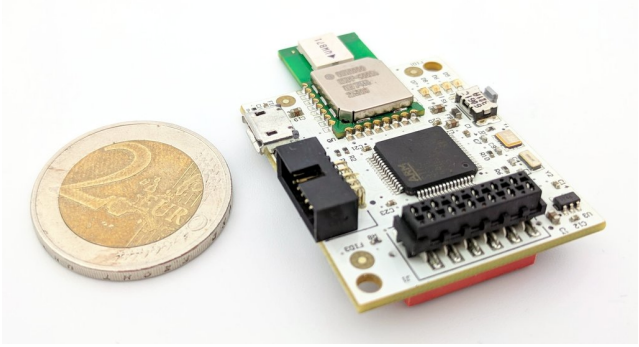

FIGURE 2. IR-UWB board with DWM1000.

The DWM1000 IR-UWB module is combined with a powerful STM32F103 microcontroller with 512 kB FLASH and 64 kB RAM memory. It can be powered via USB, an external device or a battery. The data connection between the UWB node and the experimentation platform can be established over a USB virtual serial port or USART interface accessible on an extension connector.

In the measurement campaign, two UWB nodes are connected over USB to two separate single-board computers (SBC) with a GNU/Linux operating system and from there to the experimentation workstation via WiFi or Ethernet network, as illustrated in Fig. 3. The MQTT transfer protocol is used for node control and collecting measurements to keep the measurement application complexity at the lowest level possible.

B. NLOS CLASSIFICATION DATASET

To build a CNN-based classification model for LoS and NLoS channel separation, an extensive dataset representing both classes is needed. Because each indoor environment is different in terms of multipath propagation characteristics, measurements in several different indoor environments have been collected to construct a more generic model and prevent overfitting to the specific environment.

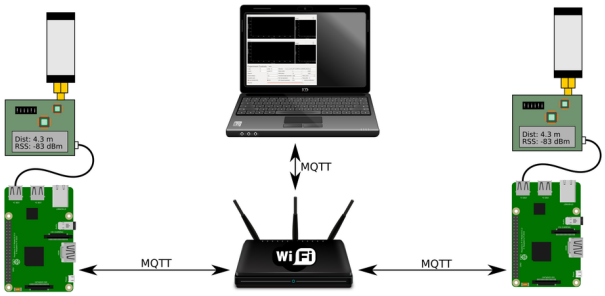

FIGURE 3. Measurement and localization system architecture.

It is very time consuming to build an extensive dataset with predefined node locations in many different indoor environments. Thus, to accelerate the process of dataset acquisition for classification, we omitted the strict node positioning on a predefined grid. Instead, one UWB node was placed at a random fixed position in a selected indoor environment, and measurements were taken while moving the other UWB node throughout the environment. First, 3000 measurements were collected for LOS channel conditions, and then 3000 measurements were collected for NLoS conditions in the same environment. To ensure that the collected measurements were always in accordance with the desired channel state, a human operator tracked the visual link between the antennas of both UWB nodes for LoS channel conditions and ensured that there was no visual link between antennas for the NLoS channel condition measurements.

In the dataset for channel condition classification, we considered 7 different indoor environments: two office environments, a small workshop, two apartments, a kitchen with a dining room and a boiler room. Each of these environments has specific multipath propagation characteristics, which add richness to the dataset and prevents the resulting model for LoS and NLoS classification to be overfitted to a single environment.

In the dataset, each measurement point includes the range estimate (based on an estimated ToF), RSS value, noise level and CIR.

C. LOCALIZATION DATASET

To test the performance of localization error mitigation algorithms, a dataset with measured ranges and actual distances between nodes and anchors should be built. From known and estimated distances between the nodes, the ranging errors are calculated and later used to build a ranging error regression model for localization error mitigation.

To this end, we defined a grid with 1 m spacing between points in two office environments. One node was placed at a random position on a grid, and the second node was successively positioned at different positions on the grid. For each grid point, 100 measurements were taken. Measurements were later separated into two different groups according to

[1]DWM1000 module, https://www.decawave.com/products/dwm1000-module

the office environments that they were taken in. One group was used as training data, and the second group was used as test data to evaluate error mitigation performance without environment overfitting.

Each data point in the dataset includes the x- and y-positions of a node, x- and y-positions of an anchor, range estimate based on an estimated ToF, RSS value, noise level and accumulated CIR envelope.

V. NODE LOCALIZATION

For a node location estimation in range-based localization systems without prior knowledge of the environment geometry, we need several anchor nodes with known locations. For a 2D localization, we need three anchors, and for 3D localization, we need at least 4 anchors within a node's range. Several estimation algorithms have been investigated in the literature, but for simplicity and a non-probabilistic approach, we selected least squares (LS) and weighted least squares (WLS) algorithms.

A. LEAST SQUARES LOCATION ESTIMATION

The distance between an anchor and a node with an unknown location is formulated as a Euclidean distance

$$d_i = \|\hat{\mathbf{p}}_e - \mathbf{p_i}\| \tag{1}$$

where d_i is the measured range between the node and the i-th anchor, $\hat{\mathbf{p}}_e$ is an unknown estimated position of a node, and $\mathbf{p_i}$ is a position of the i-th anchor.

Following the derivation provided in Appendix A, a predefined system of linear equations in a matrix form can be written, and the estimated position $\hat{\boldsymbol{\theta}}$ of a node can be calculated using an LS estimator as defined in (2)

$$\hat{\boldsymbol{\theta}}_{LS} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x}, \tag{2}$$

where $\mathbf{H}$ is an observation matrix and $\mathbf{x}$ is a vector of observed data.

B. WEIGHTED LEAST SQUARES LOCATION ESTIMATION

With ordinary least squares, the implicit assumption is having constant variances of ranging errors. However, NLoS ranges typically have increased ranging errors, as shown in Fig. 4. In fact, ranging errors increase with the severity of NLoS condition and increasing propagation distance. The contribution of NLoS ranges to the localization precision is destructive, and weighting of individual measured ranges according to the expected ranging error can help emphasize the contributions of ranges that are more reliable.

With the introduction of diagonal *NxN* positive definite weighting matrix $\mathbf{W}$ (3)

$$\mathbf{W} = \begin{bmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & w_n \end{bmatrix}, \tag{3}$$

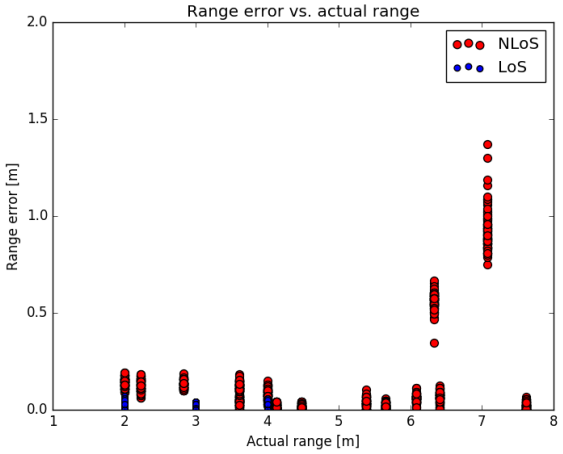

FIGURE 4. Figure shows the ranging error for different ranges and different NLoS and LoS situations.

the estimated position $\hat{\boldsymbol{\theta}}$ of a node can be expressed with the WLS estimator derived in Appendix B and defined in (4)

$$\hat{\boldsymbol{\theta}}_{WLS} = (\mathbf{H^T W H})^{-1}\mathbf{H^T W x}. \tag{4}$$

By introducing weighting matrix $\mathbf{W}$, the estimated ranging error for an individual range can be multiplied by an estimated range to favorably increase the weighting of closer anchors. The weights for a weighting matrix can be calculated by the inverse of the product of estimated range d_i and squared estimate of ranging error ϵ_i (5)

$$w_i = \frac{1}{d_i {\epsilon_i}^2}. \tag{5}$$

VI. NLOS CLASSIFICATION AND RANGING ERROR ESTIMATION USING CNN

For the NLoS classification model and ranging error regression model, we decided to use CNN due to its superior input shift invariance and ability to learn complex models. These characteristics make it possible to omit the need for preprocessing of input data and deriving low-dimensional input vectors from CIR. The proposed classification and ranging error regression CNN structures work on raw CIR data obtained directly from DW1000 IR-UWB radio in real-time localization scenarios. We implemented CNN structures using an open source software library for numerical computation using data flow graphs, namely, TensorFlow™ [15]. Its implementation is very efficient; thus, it can also run complex CNN structures in real time on computationally restricted devices such as low-end smartphones or small ARM-based single-board computers (SBCs), e.g., Raspberry Pi, BeagleBone, and so forth.

The main contribution of this work is the way in which the CNN is used. Traditionally, CNNs are fed 2-dimensional input samples of images. We reduced the dimensionality of the CNN to accept 1-dimensional input traces in the form of CIR data. The computational burden for learning and

inception processes is thus much lower compared to their 2-dimensional counterparts.

A. CONVOLUTIONAL NEURAL NETWORK STRUCTURE

The CNN is organized in a layered structure. Each unit in a single convolutional layer has its own local receptive field (a dedicated slice) of an input. It consists of several individual artificial neurons, vertically organized in planes with different sets of weights to perform multiple feature extractions on identical input. The weights of one unit are shared with all units in a layer. All units from the layer generate a set of outputs called a feature map, which is fed to the next layer. A generic layered structure of an example CNN is depicted in Fig. 5.

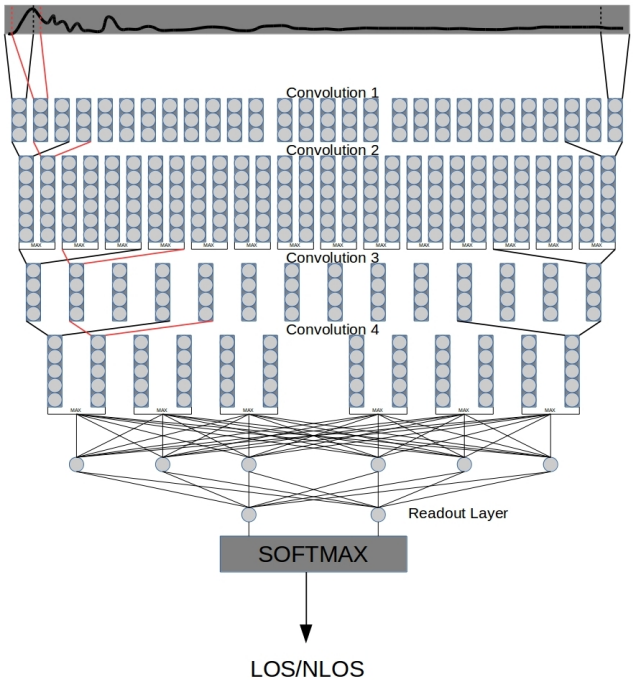

FIGURE 5. Graphical representation of a CNN architecture.

The activation function for each individual neuron in the proposed CNN structure is a rectified linear unit (ReLU) function. ReLU provides a good amount of non-linearity to the system, and it is easy and fast to calculate. It also does not impact the actual CNN performance by a significant margin compared to more complicated activation functions, such as logistic sigmoid function or hyperbolic tangent activation function.

One of the functional layers in a CNN is a spatial reduction layer, where the output data of the previous layer are down-sampled to minimize the effects of spatial positions of detected features. We selected the max pooling spatial reduction function, which selects the maximum value of the values covered within the current pooling window and propagates it to the next layer. The size of the pooling window defines the spatial reduction of the input. If the width of a pooling window is 2 and the step size (stride) is 2, then pooling selects the maximum of the two input values and the pooling window is shifted for two input values at a time. In such a case, the input dimension is reduced by a factor of 2. A good practice in CNNs is that the number of weights remains constant throughout the entire neural network. Following the spatial reduction layer, the number of weights can be kept constant with an adequate increase in the number of planes inside the convolutional layers [16].

After a series of convolutional and spatial reduction layers, a fully connected layer occurs. A combination of convolutional layers with spatial reduction layers work as an automatic input preprocessing unit that replaces the traditional complex feature extraction procedures. They serve as an automatic feature extraction layer providing features to the following fully connected neural layer. A fully connected layer has all the neurons connected to all outputs of the last convolutional layer. Then, a readout layer calculates the corresponding output.

In the case of NLoS classification, a readout layer consists of two fully connected neurons with a softmax regression algorithm on neuron outputs. The first neuron predicts the NLoS class, and the second neuron predicts the LoS class. The softmax regression output function transforms the numerical outputs of the readout layer neurons to binary values (the strongest output obtains a value of 1, and the other obtains a value of 0).

For the range error regression, the readout layer consists of a single neuron with a linear activation function returning the numerical value of the predicted ranging error.

B. CNN TRAINING

Training a neural network with high-dimensional parameter spaces requires efficient optimization algorithms. Objective functions are often stochastic because of internal data subsampling, dropout regularization and other sources of noise. Kingma and Ba [17] proposed a computationally efficient stochastic optimization algorithm, Adam, that requires only first-order gradients with little memory requirement, is invariant to diagonal rescaling of the gradients and is suitable for high-dimensional problems. It provides fast and reliable learning convergence that can be considerably faster than similar optimization algorithms (e.g., Adagrad, SGD, and so forth) [17].

To prevent overfitting of a neural network to at least some degree, dropout regularization is used to prevent complex co-adaptations on the training data. On each training sample, each neuron in a fully connected layer is randomly omitted from the network with a predefined probability (typically set to 0.5 [18]).

In our case, rather than exposing the entire training dataset to the trained neural network at once, smaller batches of 256 randomly chosen samples are fed to the network during consecutive learning iterations. In this way, we ensured more robust convergence with a higher model update frequency compared to the usual full batch learning algorithms.

C. FEEDING THE INPUT TO CNN

Channel impulse response data in a DW1000 CIR accumulator have 992 bins for the 16 MHz pulse repetition frequency (PRF) and 1016 samples for the 64 MHz PRF with a resolution of approximately 1 ns or precisely half a period of the 499.2 MHz fundamental frequency.

Every impulse response envelope in the accumulator starts rising at slightly different accumulator bins, and the exact relative starting CIR time is therefore variable. DW1000 detects and determines the start bin index of the signal with a built-in proprietary algorithm, and it can be easily accessed during the post-processing time.

In a CIR accumulator, approximately 152 bins hold most of the information available about the propagation characteristics in the environment. Visualization of one 152-bin-long CIR example is depicted in Fig. 6. During the learning, classification and regression processes, the first 152 CIR bins are used, starting at the first path index bin detected by an internal radio algorithm.

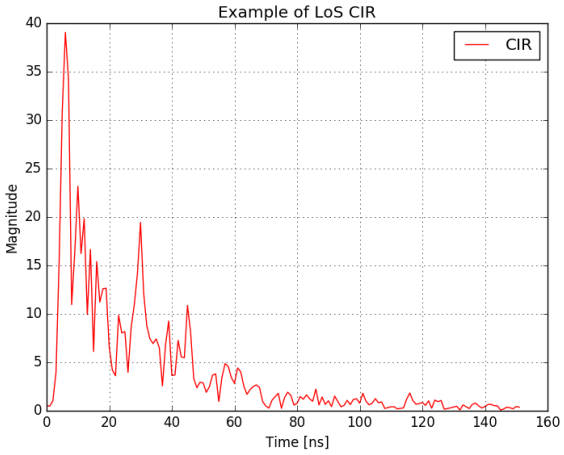

FIGURE 6. An example of LoS CIR data.

VII. PERFORMANCE EVALUATION OF CLASSIFICATION AND LOCALIZATION

Performance evaluation of the proposed CNN structures for NLoS classification and ranging error regression was performed using different subsets of data collected during the measurement campaign, as described in Sect. IV and as was used for the training of CNN models to prevent biased or overly optimistic results.

A. NLOS CLASSIFICATION PERFORMANCE COMPARISON

The proposed NLoS classification method based on CNN using raw CIR data needs to be evaluated against standard approaches where precalculated features from CIR data are used. For the classical feature-based approach, we calculated or reused readily available signal features: RSS value, first path RSS value, mean excess delay, root mean square (RMS) delay spread, kurtosis and skewness.

For the NLoS classification model based on precalculated CIR features, we used multilayer perceptron (MLP) and support vector machine (SVM) implementations from the scikit-learn Python machine learning library [19]. The SVM implementation was tested with a linear kernel function and with a radial basis function (RBF) kernel.

The MLP neural network (NN) was defined as a 3-layer MLP with one hidden layer, where the input layer has 6 neurons, the hidden layer has 152 neurons and the output layer has one neuron. The selected neural activation function was rectified linear unit (ReLU). The MLP structure was determined during an optimization process where the best performing NN candidate was selected. The structure was trained using the stochastic gradient descent (SGD) optimization algorithm with an adaptive learning rate, initial learning rate $lr = 0.1$, stopping condition $tol = 10^{-}7$, and initial network synapses set to random values.

To evaluate the performance of machine learning algorithms, some standard metrics based on the confusion matrix were calculated [20]. In the confusion matrix, rows represent the actual classes of the samples and columns represent the assigned classes within the classification process. The resulting assignments fall into four categories, called true positive (TP), true negative (TN), false positive (FP) and false negative (FN).

The applied classification performance metrics are the following:

- **Accuracy**: $\frac{TP+TN}{TP+TN+FP+FN}$ provides the percentage of correctly classified instances.
- **Precision**: $\frac{TP}{TP+FP}$ represents the percentage of correctly classified NLoS instances within all the instances that were classified as NLoS instances.
- **Sensitivity**: $\frac{TP}{TP+FN}$ is the true positive rate or a fraction of correctly classified instances within the NLoS class.
- **F1 or harmonic mean of precision and sensitivity**: $\frac{2 \cdot precision \cdot sensitivity}{precision+sensitivity}$.

The results for the four approaches are summarized in Table 1. The proposed method using CNN shows performance similar to that of standard approaches based on derived features and slightly outperforms them. The number of incorrectly classified instances is more uniformly distributed between two possible classes, which means that similar numbers of NLoS and LoS instances are incorrectly classified.

B. LOCALIZATION PERFORMANCE OF DIFFERENT APPROACHES

Next, we evaluated the localization performance of different LS and WLS approaches, comparing those with and without

TABLE 1. NLoS classification performance.

	SVM (linear)	SVM (RBF)	MLP	CNN
True positive	5825	5849	5648	5365
True negative	4085	4068	4306	5121
False positive	1915	1932	1694	879
False negative	175	151	352	635
Accuracy	82.5 %	82.6 %	82.9 %	87.4 %
Precision	75.3 %	75.2 %	76.9 %	85.9 %
Sensitivity	97.1 %	97.5 %	94.1 %	89.4 %
F1	84.8 %	84.9 %	84.7 %	87.6 %

error mitigation techniques. In all cases, error mitigation was performed using the proposed CNN-based NLoS classification and error regression models.

1) LS LOCATION ESTIMATOR WITH LOS RANGES

The best indoor localization performance is achieved when all ranges used in location estimation are LoS ranges. All LoS ranges are expected to have minimal ranging errors and thus contribute less error to the estimated location than their NLoS counterparts. LoS ranges in the experiment are filtered based on prior knowledge recorded during the measurement campaign. The results are presented in Table 2 and in Fig. 7 (LS_LOS). Similar localization performance is difficult to achieve in real-life scenarios because of an ever-changing radio environment and the absence of exact LoS information, which can only be predicted with some certainty.

TABLE 2. Accuracy of LS estimator for LoS ranges.

anchors	mean [m]	median [m]	stddev [m]
3	0.355	0.221	0.408
4	0.244	0.172	0.216
5	0.210	0.153	0.171
6	0.202	0.146	0.159
7	0.194	0.141	0.150
8	0.192	0.138	0.143
9	0.189	0.137	0.135
10	0.188	0.136	0.129

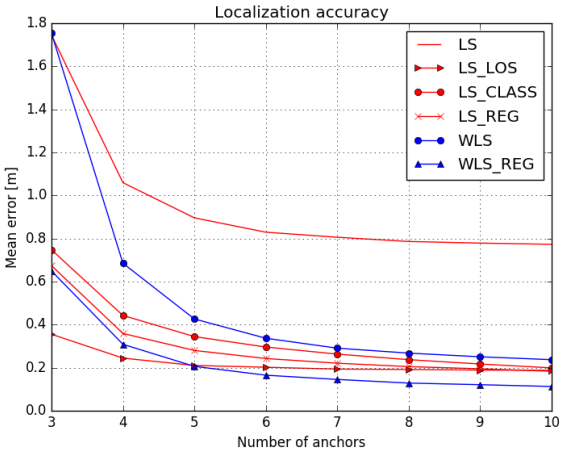

FIGURE 7. Localization accuracy for different localization algorithms and different number of anchors.

2) LS LOCATION ESTIMATOR WITHOUT ERROR MITIGATION

The LS location estimator is the simplest localization method based on range measurements from a node to several anchors. It is used when no prior knowledge is used about range measurement variances and therefore statistical assumptions about the measured data. Consequently, the expected performance greatly depends on the quality of available range measurements. In the case of NLoS ranges, range errors are high, and therefore, the expected positioning errors are also high. However, with an increasing number of anchors, the location estimation becomes refined, as depicted in Fig. 7 (LS) and summarized in Table 3.

TABLE 3. Accuracy of LS location estimator.

anchors	mean [m]	median [m]	stddev [m]
3	1.745	0.796	3.220
4	1.059	0.641	1.262
5	0.896	0.610	0.895
6	0.829	0.616	0.751
7	0.806	0.638	0.675
8	0.786	0.659	0.612
9	0.779	0.672	0.572
10	0.773	0.679	0.551

3) LS LOCATION ESTIMATOR WITH NLOS CLASSIFICATION

In this approach, CNN classification is used to detect and eliminate NLoS ranges from the localization anchor pool. The performance is lower than in the case of perfect prior NLoS knowledge under Section VII-B.1. However, such localization scenarios can only be effectively used in environments where an abundant number of LoS anchors is available. The results are presented in Table 4 and in Fig. 7 (LS_CLASS).

TABLE 4. Accuracy of LS location estimator with NLoS classification.

anchors	mean [m]	median [m]	stddev [m]
3	0.747	0.418	1.186
4	0.443	0.284	0.475
5	0.345	0.236	0.299
6	0.296	0.211	0.240
7	0.263	0.194	0.202
8	0.237	0.181	0.174
9	0.217	0.172	0.146
10	0.199	0.165	0.124

4) LS LOCATION ESTIMATOR WITH RANGING ERROR MITIGATION

In this case, ranging error is estimated for every measured range in the localization pool. Ranging error estimates are calculated based on CIR and ranging error information available in the localization dataset using the CNN model. The estimated error is subtracted from the measured range, and the resulting mitigated range is used in the LS location estimator. The results are depicted in Fig. 7 (LS_REG) and in Table 5.

TABLE 5. Accuracy of LS location estimator with ranging error mitigation.

anchors	mean [m]	median [m]	stddev [m]
3	0.674	0.237	1.452
4	0.359	0.163	0.544
5	0.280	0.137	0.373
6	0.243	0.128	0.288
7	0.221	0.120	0.249
8	0.205	0.118	0.219
9	0.195	0.111	0.208
10	0.184	0.107	0.186

5) WLS LOCATION ESTIMATOR

The information from the CNN ranging error regressor can be used as a weighting factor in WLS location estimation. As shown in Fig. 7 (WLS) and in Table 6, much greater accuracy can be achieved than with the bare LS estimator as the number of available anchors increases.

TABLE 6. Accuracy of WLS location estimator.

anchors	mean [m]	median [m]	stddev [m]
3	1.754	0.792	3.384
4	0.685	0.407	0.982
5	0.426	0.304	0.457
6	0.337	0.256	0.305
7	0.291	0.230	0.239
8	0.268	0.216	0.207
9	0.251	0.207	0.183
10	0.237	0.200	0.164

6) WLS LOCATION ESTIMATOR WITH RANGING ERROR MITIGATION

The combination of WLS location estimator and ranging error mitigation with CNN ranging error regression shows the best localization performance of all methods without filtering the NLoS ranges out. The results are shown in Table 7 and in Fig. 7 (WLS_REG).

TABLE 7. Accuracy of WLS location estimator with ranging error mitigation.

anchors	mean [m]	median [m]	stddev [m]
3	0.650	0.232	1.330
4	0.309	0.126	0.516
5	0.206	0.090	0.303
6	0.165	0.071	0.231
7	0.145	0.060	0.199
8	0.129	0.053	0.172
9	0.121	0.048	0.165
10	0.113	0.045	0.149

VIII. COMPUTATIONAL PERFORMANCE EVALUATION

As described in Section VI, we selected TensorFlow for the implementation of CNN since it performs well on various computing platforms, from workstation PCs, servers or clusters to low-end smartphones and SBCs running GNU/Linux OSs. In fact, we were particularly interested in the possibility of implementing and operating CNN-based algorithms in computationally restricted devices. TensorFlow Lite[2] has recently been released and is specially designed and optimized to run on smartphones with as few resources as possible. Rather than using the standard Protocol Buffers[3] for structured data serialization, FlatBuffers[4] data serialization is introduced with a code footprint that is an order of magnitude smaller and no need for parsing to access data. There is also a previous version of mobile implementation, TensorFlow Mobile[5], but both implementations support only mobile platforms such as Android or iOS. On GNU/Linux devices, only the standard TensorFlow distribution can be used to the best of our knowledge.

A. COMPUTING PLATFORMS UNDER TEST

For testing and performance evaluation of the proposed CNN-based algorithms, we selected 6 broadly accessible and widespread computing platforms. We classified them into two distinct groups: restricted capability and performance line platforms. Restricted capability platforms are intended for use in distributed sensor nodes, remote user terminals or edge gateways. Performance line platforms are intended for use in computationally intensive, centralized and cloud applications.

One of the most popular restricted capability and low-budget computing platforms is Raspberry Pi[6] running the Debian-based Raspbian[7] GNU/Linux operating system. All versions of Raspberry Pi use ARM microprocessors similar to those found in smartphones. In the evaluation, we used the version 1 (RPI1) with a single-core ARM1176JZF-S microprocessor, 512 MB of RAM and 700 MHz clock, as well as the latest and most powerful version 3 (RPI3) with a quad-core ARM Cortex-A53 CPU, 1 GB of RAM and 1.2 GHz CPU frequency. The third restricted capability computing platform is represented by a netbook (NET) from Asus based on a 64-bit x86 Intel Atom N450 processor with two threads, 2 GB of RAM and 1.66 GHz base CPU frequency.

Performance line computing platforms are represented by one ultrabook computer and two high-performance workstation laptops. The Lenovo Yoga ultrabook (UB) is based on a 64-bit Intel i7-4500U CPU with 4 threads, 1.8 GHz base CPU frequency and 8 GB of RAM. The remaining high-performance platforms are an HP EliteBook 8560w (WS1) based on an Intel i7-2670QM CPU with 8 threads and 8 GB of RAM and a Dell XPS15 (WS2) based on an Intel i7-6700HQ CPU with 8 threads and 16 GB of RAM. All of the selected platforms run on identical Ubuntu 16.04.3 LTS GNU/Linux OS except the Raspberry Pi platforms running the Debian 9.1 GNU/Linux-based Raspbian OS. The parameters of the selected platforms are collected in Table 8.

B. PERFORMANCE TEST DEFINITION

We developed a specific performance test to evaluate how fast a platform with the proposed CNN-based NLoS channel classification model can classify input samples according to the size of the sample batch. TensorFlow is optimized for numerical computations on large organized multidimensional

[2]TensorFlow Lite, https://www.tensorflow.org/mobile/tflite/

[3]Protocol Buffers, https://developers.google.com/protocol-buffers/

[4]FlatBuffers, https://google.github.io/flatbuffers/

[5]TensorFlow Mobile, https://www.tensorflow.org/mobile/

[6]Raspberry Pi, https://www.raspberrypi.org/

[7]Raspbian, https://www.raspberrypi.org/downloads/raspbian/

TABLE 8. Computing platforms used in performance evaluation.

	RPI1	NET	RPI3
CPU	ARM1176JZF-S	Atom N450	ARM Cortex-A53
Clock [GHz]	0.7	1.66	1.2
Threads	1	2	4
RAM	512 MB	2 GB	1 GB
OS	Debian 9.1	Ubuntu 16.04	Debian 9.1
TF ver.	1.4.0	1.4.0	1.4.0
	UB	WS1	WS2
CPU	i7-4500U	i7-2670QM	i7-6700HQ
Clock [GHz]	1.8 - 3.0	2.2 - 3.1	2.6 - 3.5
Threads	4	8	8
RAM	8 GB	8 GB	16 GB
OS	Ubuntu 16.04	Ubuntu 16.04	Ubuntu 16.04
TF ver.	1.4.0	1.4.0	1.4.0

arrays of numerical values (tensors) and is expected to work best with larger batches of input samples. The test is written in the Python programming language (version 2.7) using the Python extension libraries NumPy, scikit-learn [19] and Pandas.

Algorithm 1 Performance Test Algorithm

```
data_set ← load_data

for i = 0 to 11 do
    start_time ← current_time
    for j = 1 to 1000 do
        classify(data_set.getNextBatch(2^i))
    end for
    end_time ← current_time
    performance[i] ← (1000*2^i)/(end_time−start_time)
end for
```

The test script is presented in pseudocode in Algorithm1. The algorithm loads and initializes the classification model inside the TensorFlow environment. It then loads the test data from .csv files and formats them according to the requirements defined during the NLOS classification model training stage. First, 152 bins from the indicated first path index inside the individual range estimate CIR accumulator are taken and added to the test dataset array. Next, a test algorithm measures the time for executing 1000 classification iterations with successively increasing sample batch sizes. Sample batch sizes increase with 2^n, where n goes from 0 to 11, i.e., holding from 1 to 2048 samples.

During the classification procedure, each CIR sample is first centered and scaled by removing the mean and scaling to unit variance element-wise based on relevant statistics computed on the training dataset. The standardized sample batch is then fed to the TensorFlow implementation of the trained CNN NLoS classification model. The number of classified samples during the 1000 iterations is divided by the elapsed computation time to obtain the classification performance of a platform under test.

C. PERFORMANCE RESULTS

The performance requirements for computing platforms depend on the position that they are used at in a network. In the case of a UWB localization system exploiting CIR information, an end device only requires the capability to process a few or at maximum few tens of CIR samples per second, and an edge device such as an anchor node requires few hundreds of samples per second to provide sufficient localization refresh rates. When localization is provisioned on a centralized infrastructure level, a central computing platform must provide several thousands of processed links and also be able to process other computing tasks.

All tested platforms present similar behavior regarding the sample batch sizes. The performance increases with increasing batch size and stops when computing resources are optimally utilized. TensorFlow is designed to work optimally with larger chunks of data, which is evident from the performance graphs depicted in Fig. 8. The performance with small batch sizes is limited by the data loading speed and the response times from individual components in the computational chain (loading data, data scaling, and classification). With larger sample batches, there are less data handling procedures per sample, and with multithreaded CPUs, the workload can be distributed more efficiently. Most of the tested platforms achieve peak performance with batches at 1024 samples, except for the least powerful platforms RPI1 and NET. These platforms achieve the peak performance earlier because of their smaller memory size, lower number of CPU threads and lower CPU performance.

The RPI1 platform is able to process 92 classifications per second at a batch size of 4 samples. That is sufficient for processing at an end device level, satisfying the minimum requirement for 3D localization. The maximum performance achieved with the RPI1 platform is 212 classifications per second with a batch size of 256 samples. The newer and improved RPI3 platform is powerful enough to run as an edge device with 514 classifications per second with a batch size of 4 samples, 3577 classifications per second with a batch size of 256 samples and achieves maximum performance of 4252 classifications per second with a batch size of 2048 samples. The NET platform is also suitable for an edge device with a peak performance of 1688 classification per second at a batch size of 2048 samples but is an outdated version of a platform with a higher price than RPI1 and RPI3.

On the high-performance side of the tested platforms, suitable for centralized localization solutions, the entry-level UB platform achieves 3723 classifications per second with a batch size of 4 samples, 17339 classifications per second with a batch size of 256 samples and achieves peak performance of 18183 classifications per second with a batch size of 512 samples. The high-performance WS1 platform achieves 1223 classifications per second with a batch size of 4 samples, 29182 classifications per second for a batch size of 256 samples and achieves maximum performance of 34933 classifications per second with a batch

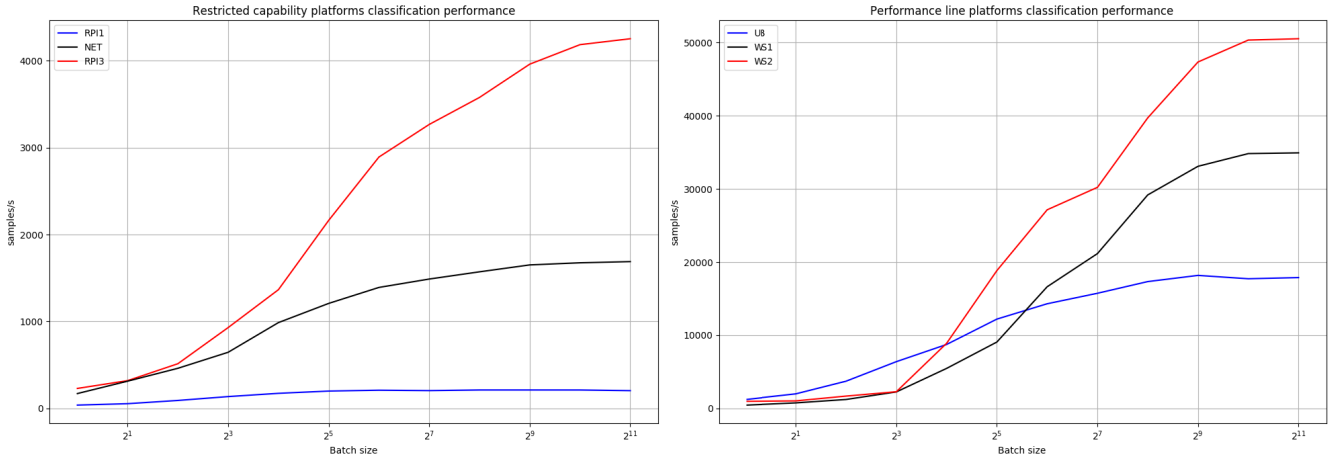

FIGURE 8. Classification performance of different hardware platforms.

size of 2048 samples. The more recent workstation laptop WS2 achieves 1686 classifications per second with a batch size of 4 samples, 39713 classifications per second at a batch size of 256 samples and maximum performance of 50531 samples per second with a batch size of 2048 samples.

The results presented above confirm that in a distributed localization system, the proposed CNN-based NLoS classification algorithm using TensorFlow can also be utilized on restricted capability end and/or edge devices. Thus, compared to the centralized implementation, the implementation that utilizes computing in end and edge devices

- avoids the need to transfer large amounts of CIR data to the computing platforms on centralized localization servers or in the cloud,
- achieves lower latency in localization, which is essential in several application domains, and
- guarantees greater scalability.

IX. CONCLUSION

In this paper, a novel approach to indoor localization is proposed and evaluated using two CNN-based methods working on raw UWB CIR data. In particular, the two methods use NLoS channel classification and ranging error regression models for reducing the error in indoor localization without any prior radio environment knowledge. The performance comparison of NLoS channel classification confirms that the method with CNN and raw CIR data slightly outperforms the methods based on statistically derived CIR features. When used with LS and WLS location estimators as techniques for localization error mitigation, the proposed CNN-based NLoS classification and error regression significantly improve the localization performance of the baseline LS and WLS methods. Ranging error regression used on all available ranges performs better than filtering out NLoS ranges. As part of future work, additional extensive measurement campaigns are planned in different indoor environments to capture further LoS and NLoS propagation conditions and subsequently build more general NLoS classification and ranging error regression models for further validation.

While providing more reliable indoor localization results, the newly proposed methods imply higher input data dimensionality, which combined with CNN complexity yields higher computing demands compared to classical approaches. However, the computational performance evaluation on various computing platforms indicates that they satisfy the requirements of the TensorFlow implementation of the methods with an appropriately restricted scope of localization. Thus, the proposed methods can improve indoor localization in both centralized and distributed deployments, the latter requiring network topology optimization while providing a good trade off in terms of required bandwidth for transferring CIR data and enabling lower latency.

APPENDIX A
LEAST SQUARES LOCATION ESTIMATOR

The LS estimator chooses θ such that it makes a signal model $s[n]$ closest to the observed data $x[n]$. The LS error criterion is defined by (A.1). The value of θ that minimizes $J(\theta)$ is the LS estimator.

$$J(\theta) = \sum_{n=0}^{N-1} (x[n] - s[n])^2 \tag{A.1}$$

For 2-D localization, the Euclidean distance between an anchor and a node is formulated as in (A.2)

$$d_i = \sqrt{(x_i - x)^2 + (y_i - y)^2} \tag{A.2}$$

where d_i is the measured range between the node and the i-th anchor, x_i and y_i are coordinates of the i-th anchor, and x and y are the coordinates of a node that are not yet known.

Applying the linear LS approach for a scalar parameter, we assume (A.3), where $h[n]$ is a known sequence.

$$s[n] = \theta h[n] \tag{A.3}$$

The LS error criterion is now transformed into the form in (A.4).

$$J(\theta) = \sum_{n=0}^{N-1} (x[n] - \theta h[n])^2 \tag{A.4}$$

For (A.3), the matrix notation in (A.5) can be used for the signal $\mathbf{s} = [s[0]s[1] \dots s[N-1]]^T$, where $\mathbf{H}$ is a known $N \times p$ observation matrix of full rank p.

$$\mathbf{s} = \mathbf{H}\boldsymbol{\theta} \tag{A.5}$$

The LS estimator can be found by minimizing (A.6), resulting in (A.7).

$$\begin{aligned} J(\theta) &= \sum_{n=0}^{N-1} (x[n] - s[n])^2 \\ &= (\mathbf{x} - \mathbf{H}\boldsymbol{\theta})^T (\mathbf{x} - \mathbf{H}\boldsymbol{\theta}) \qquad \text{(A.6)} \\ \hat{\boldsymbol{\theta}} &= (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x} \qquad \text{(A.7)} \end{aligned}$$

Rewriting (A.2) provides (A.8).

$$-2x_i x - 2y_i y + x^2 + y^2 = \hat{d}_i^2 - x_i^2 - y_i^2 \tag{A.8}$$

To linearize the system of non-linear equations, we introduce a new variable in (A.9)

$$R = x^2 + y^2 \tag{A.9}$$

and define the estimated location with (A.10).

$$\hat{\boldsymbol{\theta}} = \begin{bmatrix} x & y & R \end{bmatrix}^T \tag{A.10}$$

We can now write the system of equations in a matrix form

$$\mathbf{H} = \begin{bmatrix} -2\,x_1 & -2\,y_1 & 1 \\ -2\,x_2 & -2\,y_2 & 1 \\ \vdots & \vdots & \vdots \\ -2\,x_N & -2\,y_N & 1 \end{bmatrix} \tag{A.11}$$

$$\mathbf{x} = \begin{bmatrix} \hat{d}_1^2 - x_1^2 - y_1^2 \\ \hat{d}_2^2 - x_2^2 - y_2^2 \\ \vdots \\ \hat{d}_N^2 - x_N^2 - y_N^2 \end{bmatrix} \tag{A.12}$$

which provides us a predefined system of linear equations in a normal form where position can be calculated using the LS estimator defined in Eq. (A.7).

APPENDIX B
WEIGHTED LEAST SQUARES LOCATION ESTIMATOR

With the introduction of diagonal *NxN* positive definite weighting matrix $\mathbf{W}$ (B.1)

$$\mathbf{W} = \begin{bmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & w_n \end{bmatrix}, \tag{B.1}$$

the LS estimation error given in Eq. (A.6) can be rewritten as (B.2)

$$\begin{aligned} J(\theta) &= \sum_{n=0}^{N-1} w_i (x[n] - s[n])^2 \\ &= (\mathbf{x} - \mathbf{H}\boldsymbol{\theta})^T \mathbf{W} (\mathbf{x} - \mathbf{H}\boldsymbol{\theta}). \qquad \text{(B.2)} \end{aligned}$$

The weights for a weighting matrix (B.1) can be calculated by the inverse of the product of the estimated range d_i and squared estimate of ranging error ϵ_i (B.3)

$$w_i = \frac{1}{d_i \epsilon_i^2}. \tag{B.3}$$

The estimated position $\hat{\boldsymbol{\theta}}$ of a node can now be expressed with the WLS estimator defined in (B.4)

$$\hat{\boldsymbol{\theta}}_{WLS} = (\mathbf{H^T W H})^{-1} \mathbf{H^T W x}. \tag{B.4}$$

REFERENCES

[1] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, R. L. Moses, and N. S. Correal, "Locating the nodes: Cooperative localization in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 54–69, Jul. 2005.
[2] H.-S. Ahn and W. Yu, "Environmental-adaptive RSSI-based indoor localization," *IEEE Trans. Autom. Sci. Eng.*, vol. 6, no. 4, pp. 626–633, Oct. 2009.
[3] K. Wu, J. Xiao, Y. Yi, D. Chen, X. Luo, and L. M. Ni, "CSI-based indoor localization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1300–1309, Jul. 2013.
[4] S. Yiu and K. Yang, "Gaussian process assisted fingerprinting localization," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 683–690, Oct. 2016.
[5] S. He, B. Ji, and S.-H. G. Chan, "Chameleon: Survey-free updating of a fingerprint database for indoor localization," *IEEE Pervasive Comput.*, vol. 15, no. 4, pp. 66–75, Oct. 2016.
[6] H.-J. Shao, X.-P. Zhang, and Z. Wang, "Efficient closed-form algorithms for AOA based self-localization of sensor nodes using auxiliary variables," *IEEE Trans. Signal Process.*, vol. 62, no. 10, pp. 2580–2594, May 2014.
[7] S. Gezici *et al.*, "Localization via ultra-wideband radios: A look at positioning aspects for future sensor networks," *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 70–84, Jul. 2005.
[8] D. Dardari, A. Conti, U. Ferner, A. Giorgetti, and M. Z. Win, "Ranging with ultrawide bandwidth signals in multipath environments," *Proc. IEEE*, vol. 97, no. 2, pp. 404–426, Feb. 2009.
[9] B. Silva and G. P. Hancke, "IR-UWB-based non-line-of-sight identification in harsh environments: Principles and challenges," *IEEE Trans. Ind. Informat.*, vol. 12, no. 3, pp. 1188–1195, Jun. 2016.
[10] S. Marano *et al.*, "NLOS identification and mitigation for localization based on UWB experimental data," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 7, pp. 1026–1035, Sep. 2010.
[11] Z. Xiao, H. Wen, A. Markham, N. Trigoni, P. Blunsom, and J. Frolik, "Non-line-of-sight identification and mitigation using received signal strength," *IEEE Trans. Wireless Commun.*, vol. 14, no. 3, pp. 1689–1702, Mar. 2015.
[12] T. Van Nguyen, Y. Jeong, H. Shin, and M. Z. Win, "Machine learning for wideband localization," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 7, pp. 1357–1380, Jul. 2015.
[13] H. Wymeersch, S. Marano, W. M. Gifford, and M. Z. Win, "A machine learning approach to ranging error mitigation for UWB localization," *IEEE Trans. Commun.*, vol. 60, no. 6, pp. 1719–1728, Jun. 2012.
[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[15] M. Abadi *et al.* (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [Online]. Available: https://www.tensorflow.org/
[16] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. Cambridge, MA, USA: MIT Press, 1998, pp. 255–258. [Online]. Available: http://dl.acm.org/citation.cfm?id=303568.303704
[17] D. P. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: http://arxiv.org/abs/1412.6980

[18] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. (2012). "Improving neural networks by preventing co-adaptation of feature detectors." [Online]. Available: http://arxiv.org/abs/1207.0580
[19] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011. [Online]. Available: http://hal.inria.fr/hal-00650905
[20] M. Kulin, C. Fortuna, E. De Poorter, D. Deschrijver, and I. Moerman, "Data-driven design of intelligent wireless networks: An overview and tutorial," *Sensors*, vol. 16, no. 6, p. 790, 2016. [Online]. Available: http://www.mdpi.com/1424-8220/16/6/790

KLEMEN BREGAR received the B.Sc. degree in electrical engineering from the University of Ljubljana, Ljubljana, in 2013. He is currently pursuing the Ph.D. degree in information and communication technologies with the Jožef Stefan International Postgraduate School. He is currently a Research Assistant with the Department of Communication Systems, Jožef Stefan Institute, Ljubljana. His research interests include the development of embedded systems, wireless sensor networks, biomedical sensor systems, and machine learning. His recent research interests include indoor localization, ultra-wide bandwidth communication systems, biomedical wireless sensor networks, and the application of deep learning neural networks. He is currently involved in H2020 projects eWine and SAAM and contributes to a national applied research project on advanced ray-tracing techniques in radio environment characterization and radio localization (L2-7664).

MIHAEL MOHORČIČ (M'02–SM'10) received the B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from the University of Ljubljana, Ljubljana, in 1994, 1998, and 2002, respectively, and the M.Phil. degree in electrical engineering from the University of Bradford, Bradford, U.K., in 1998. He is currently the Head of the Department of Communication Systems and a Scientific Advisor with the Jožef Stefan Institute, and an Associate Professor with the Jožef Stefan International Postgraduate School. He has authored or co-authored over 140 refereed journal and conference papers, co-authored two books and contributor to nine book chapters. His research interests include the development and performance evaluation of network protocols and architectures for mobile and wireless communication systems, and resource management in terrestrial, stratospheric, and satellite networks. His recent research interest is focused on cognitive radio networks, smart applications of wireless sensor networks, dynamic composition of communication services, and wireless experimental testbeds. He participated in many EU co-funded research projects since 1996 and is currently involved in H2020 projects eWINE, Fed4FIRE+, NRG-5, and SAAM. He also coordinates a national applied research project on advanced ray-tracing techniques in radio environment characterization and radio localization (L2-7664).