Convolutional Neural Networks for Position Estimation in TDoA-based Locating Systems

Arne Niitsoo† arne.niitsoo@iis.fraunhofer.de
Thorsten Edelhäußer† thorsten.edelhaeusser@iis.fraunhofer.de
Christopher Mutschler†‡ christopher.mutschler@iis.fraunhofer.de

†Machine Learning and Information Fusion Group
Fraunhofer Institute for Integrated Circuits IIS
Nuremberg, Germany

‡Maching Learning and Data Analytics Lab
Friedrich-Alexander University Erlangen-Nürnberg (FAU)
Erlangen, Germany

***Abstract*—Object localization and tracking is essential for many applications including logistics and industry. Many local Time-of-Flight (ToF)-based locating systems use synchronized antennas to receive radio signals emitted by mobile tags. They detect the Time-of-Arrival (TOA) of the signal at each antenna and trilaterate the position from the Time Difference-of-Arrival (TDoA) between antennas. However, in multipath scenarios it is difficult to extract the correct ToA. This causes wrong positions.**

This paper proposes a signal processing method that uses deep learning to estimate the absolute tag position directly from the raw channel impulse response (CIR) data. We use the CIR together with ground truth positional data to train a convolutional neural network (CNN) that not only estimates non-linearities in the signal propagation space but also analyzes the signal for multipath effects. Our evaluation shows that our position estimation works in multipath environments and also outperforms classical signal processing in line-of-sight situations.

I. Introduction

The positional tracking of people, material, and tools in industrial environments is considered as a key component to digitalization. But vision-based tracking with its high positional accuracy is no viable solution as it does not guarantee robust tracking. Occlusion prevents a continuous tracking.

Radio-based real-time locating systems (RTLSs) do not suffer from occlusion that much. However, to make them work in practice is also tedious. Besides costs for hardware, software, and system installation it is often required to tune the system for the target environment. This often requires a field campaign to manually optimize the system parameters for the desired application. Non-experts usually cannot install RTLSs. This causes high costs, which prevents their application.

In growth markets and in environments that are typical for industrial applications (production and machinery buildings, logistics centers etc.) there is also a big challenge for radio-based RTLSs in terms of robustness and reliability. Metallic surfaces often cause signal reflections and attenuations, which favor multipath signal propagation that leads to a wrong position. A solution is to install the antennas at positions such that multipath becomes rare. However, this is a highly non-linear optimization problem that often still results in poor configurations. A common approach is to install more antennas than necessary. However, this causes high costs.

There is a multitude of approaches that either deal with multipath propagation, i.e., unscented Kalman-filters [1], channel classification [2], subsample interpolation [3], or sub-space approaches [4], or that even exploit the presence of multipath, i.e., by generating scattering models with statistics [5], [6], by simultaneous target and multipath positioning [7], by using training signals to model a random variable [8] together with a floor plan to enhance the tracking filter [9] or using large-scale MIMO [10]. However, those approaches either do not scale well or do not improve with an increase of available training signals (as those can often be acquired easily).

The key idea is to use machine learning (ML) instead of traditional signal processing to estimate positions. Deep learning (DL) has shown to outperform classical approaches in several applications and has also been used for localization purposes [11]. DL extracts relevant features embedded in the signals itself. This paper uses the channel impulse response (CIR) from the antennas together with ground truth positional data (derived by a robot equipped with an optical reference system) to train a deep convolutional neural network (CNN). The CNN models both the linear and multipath propagation of the environment. Once it has been trained we can also fine-tune the CNN for different (multipath) environments.

The paper is structured as follows. Sec. II reviews related work before Sec. III briefly covers background on signal processing in RF-localization systems and (convolutional) neural networks. Next, Sec. IV shows how to apply CIRs to convolutional neural networks for position estimation in TDoA-based systems, i.e., data (pre-)processing and normalization schemes. Sec. V describes the experimental setup and the datasets. Sec. VI evaluates our position estimator on different real world datasets and shows that we outperform conventional position estimators not only in massive multipath environments but that it can also compete in line-of-sight (LOS) scenarios.

II. Related Work

There is much work that uses the received signal strength (RSS) [12]–[14], the time of arrival (ToA) [15]–[17], or their combinations [18]. Some use ML-based schemes such as neural networks with a single hidden layer [13], [16], [18], variants of neural networks (i.e, deep belief networks [14],

978-1-5386-5635-8/18/$31.00 ©2018 IEEE

deep neural networks [19], fuzzy neural networks [20], artificial synaptic networks [15]), Gaussian regression [21], support vector machines (SVM) [17], or combinations of them [12]. Iqbal et al. [22] monitor patients using CNNs to correlate RSS measurement in a clinical environment. However, all these methods only use RSS- or ToA-features, or combinations, which are only rough features in a multipath environment.

A much richer feature to estimate the location of an object is the channel impulse response (CIR). Yu et al. [23] extract energy and delay features from the UWB impulse response and use these features to train a neural network. Li et al. [24] extract features from [25] to identify LoS (NLoS) situations using an SVM. Cui et al. [26] use a neural network to approximate the relationship between the SNR and statistical information such as skewness and kurtosis in the CIR. Savic et al. [27] propose a kernel-PCA combined with Gaussian process regression that projects the channel parameters onto a nonlinear space from which then a subset is used for ranging. Ergut et al. [28] use a set of anchors to generate multipath profiles, i.e., a number of time differences between peaks within a single CIR, which are used together with ground truth data to train a neural network with a single hidden layer. Jin et al. [29] approximate the CIR from subcarrier amplitudes of OFDM signals and propose a fingerprinting based on Gaussian regression. Also known as channel state information (CSI) this has extensively been studied lately [30]. However, all of the above approaches extract hand-crafted features from the CIR. Those only represent a subset of the available information, which only results in a rough estimation.

In contrast to a manual feature extraction deep learning (DL) aims at finding and extracting the relevant features from the sensor data directly. This requires more data. Wang et al. [31]–[34] propose several ideas to process the CSI from WiFi OFDM-signals using deep CNNs. They feed the CSI directly into a CNN to train a position [31], train with phase information [32], directly estimate the angle of arrival with a CNN using phase fingerprinting [33], and combine these ideas [34]. However, their difference lies in the nature of the underlying signals and the system setup. TDoA-based localization requires a synchronized network of access points, i.e., anchors. Different from ToA the subcarrier amplitudes describe the signal propagation profile and hence the relationship between access point and mobile devices at specific positions.

Tiemann et al. [35] use DL to estimate orientation-dependent error induction characteristics from the CIR. However, they do not consider CIRs from synchronized antennas and do not estimate the position. Vieira et al. [36] use convolutional neural networks to fingerprint massive MIMO channels. However, the signals and the system setup in massive MIMO channel fingerprinting are significantly different. Comiter et al. [37] propose a beam estimation for using deep neural networks that derives the angle of arrival by phase differences. Using different antenna arrays a structured pair of neural networks is used to estimate the antenna beam. However, they also do not use several CIRs to estimate the position within a ToA-setup. Xiao et al. [38] propose denoising autoencoders to model the noise of reference locations. In the localization phase the measurement point is denoised by the autoencoder and a k-Nearest-Neighbor (KNN) classifier estimates the location. However, both the setup and the signals significantly differ from our approach.

III. Background

A. Position Estimation and Channel Impulse Response (CIR)

RTLSs that use time difference of arrival (TDoA) estimation need synchronized receivers and estimate not only the position of the mobile tag but also its time-of-transmission (ToT). Both result from a hyperbolic trilateration of the TDoA-values.

We derive the time of arrival (ToA) at each antenna through an analysis of the channel impulse response (CIR), see Fig. 1. The blue signal shows the CIR under line-of-sight conditions. We can extract the ToA by estimating the peak in the signal (at this point the correlation between the measured signal and the waveform is at its maximum), which is 30ns from the window start in Fig. 1. But although the signal is clear we can only extract the real ToA with a bias that is introduced by the bandwidth-limited analog signal. Clever interpolation between sampling points and inclination point ToA-estimation [1] help to reduce this bias to a minimum.

However, not only a limited sampling frequency adds a bias. Consider the red CIR in Fig. 1. Due to multipath propagation the signal travels along many routes until it reaches the antenna. This makes the ToA estimation ambiguous. There are a number of ToA estimators that make use of thresholds, maximum energy, and crossings to determine the correct ToA [26]. However, in multipath situations the real ToA is still often unknown and cannot be estimated from the CIR alone without further information. Apparently, such ToA errors cause incorrect TDoAs, which lead to a large bias in the position.

B. (Convolutional) Neural Networks

Artificial neural networks consist of many interconnected simple units, i.e., neurons. Classic feed-forward networks have layers of fully connected neurons, see the two last layers in Fig. 2. The layers that we cannot see from the outside, i.e., any layers in between the input and the output layer, are hidden layers. With data at the input layer the neurons propagate activations and provide information at the output layer. Artificial neural networks are generalized function approximators and

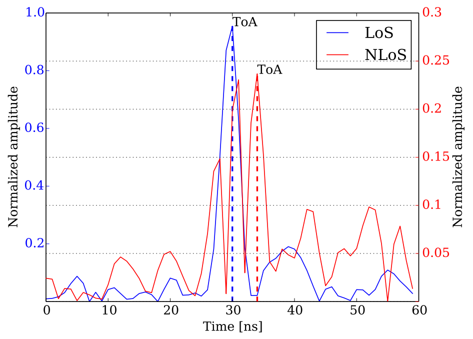


Fig. 1. Channel impulse response (CIR).

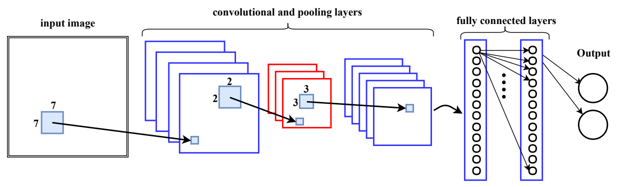


Fig. 2. Principle architecture of a convolutional neural network.

their depth (number of hidden layers) defines the complexity of the functions they can approximate.

Each connection to a neuron has an assigned weight parameter w that controls the influence of the preceding neuron. A connection of the i-th neuron of layer k with the j-th neuron of layer $k+1$ is defined by a weight w_{ij}^k. To calculate the neuron activation throughout the network we iteratively calculate the output $h(j)$ of any single neuron j per layer $k>0$

$$h_j^k(\mathbf{x}) = g(b_j^k + \sum_{i=0}^{n} w_{ij}^k x_i^{(k-1)}),$$

where b_j^k is a bias parameter, w_{ij}^k is the weight of the neuron connection, $x_i^{(k-1)}$ is the activation of the previous neuron, and $g(\cdot)$ is a (non-linear) activation function, such as a sigmoid function or the rectifier linear unit (ReLU). In practice the weights are shared in a matrix and the network is calculated using matrix multiplications along the layers.

Finding an optimal set of network parameters poses a non-convex optimization problem that is hard to solve mathematically. But gradient-based optimization, such as stochastic gradient descent (SGD) works well in practice. Given a labeled data set (where both the input and the output labels are known), we can use the data as input, calculate the neuron activations layer by layer and read the output of the network. On the output we apply a loss-function, e.g. mean-square error, that defines how good a network approximated the label of the data. Next, we back-propagate the prediction error through the network, i.e., we calculate the influence of each neuron (activation) on the total error and use the gradients at each neuron as an indicator to reduce its influence on the error. Finally, we update the weight parameters using a (small) learning rate such that the loss decreases in future predictions.

After successful termination of the learning process, i.e., after some fixed number of iterations or by a threshold on the calculated loss, we determine the classification performance by applying test data to the neural network. If the performance values meet the desired criteria, the process of training a neural network is completed and it is ready to classify new data.

Convolutional neural networks define a special architecture of neural networks. They use pooling layers and normalizations layers interchangeably between consecutive convolutional layers, see Fig. 2.

The convolutional layers apply a convolution operation to the input (often the input is an image) to extract the features that are embedded in the training set. Each convolutional layer is also followed by a (non-linear) activation function. A convolution is a filter operator that is (conceptually) slid over the input and that preserves the spatial relationship between the input data points. Usually, we apply several convolutional filters. For instance, a 7×7 convolution filter next to 3 more run directly on the input image in Fig. 2.

The pooling layer down-samples the data. Popular pooling operations are max- and average-pooling. The intuition behind them is that once the convolution layer has learned the features from the underlying data the pooling kernels (in Fig. 2 with size of 3) run over the feature maps and keep the activations according to the pooling policy. This keeps the information while it reduces the spatial dimension and computation time.

Usually the first layers of a CNN look at low level features such as edges and curves. As we stack more convolutional layers on top the features become more sophisticated and extract more advanced features. The fully connected layers at the end are used for classification.

IV. Data Preparation

A. Data Preprocessing

Machine learning methods take training data to build up a model that can later be used to predict the value of a previously unseen data point. However, in TDoA-based systems the transmitter is not time-synchronized, i.e., the time-of-transmit t_{ToT} is unknown. Hence, we also do not know at which time t_{CIR}^i a channel impulse response at an antenna i actually starts. Often, a simple triggering method, e.g. a threshold method, is used to set the start of the impulse response. However, this poses two challenges. First, a single CIR only contains *relative* timing information (including its ToA, e.g. the peak in Fig. 3 for line-of-sight) in relation to the window start time t_{CIR}. Second, t_{CIR} has no common timing across the receiving antenna channels i, due to varying analog signal processing delays, different lengths of antenna cables, and timing jitter of internal clocks. All of them are also affected by temperature changes. Hence, a set of CIRs originating from the same mobile tag position also differs over time.

However, a simple calibration of timing offsets that normalizes t_{CIR}^i helps to compensate for that. For this purpose, we derive a relative calibration offset Δt^i per antenna unit i (with one antenna arbitrarily but statically being set to 0) using reference transmitters at known locations (see the green time

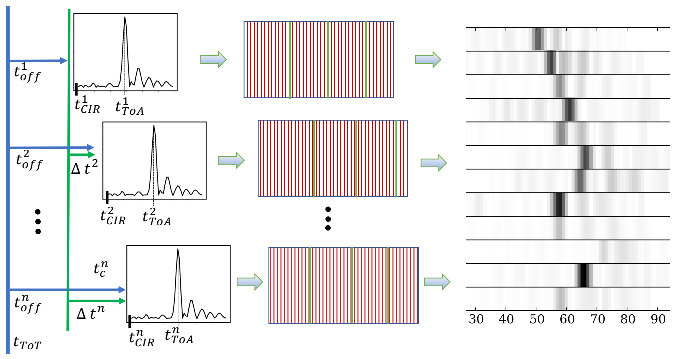


Fig. 3. CIR, calibration padding.

offset in Fig. 3).[1] This converts the ToA-based estimation into a TDoA-based estimation, as we only use the differences to a single antenna from now. We use these values to pre-process t^i_{CIR} such that all the CIRs together are free of timing-offsets and hence stable over time. However, although this also adds a bias to the position (as there is a bias for the ToA estimation of the reference transmitters) it works fine in practice.

Now consider the CIR to be a time-discrete series of values that have been sampled with a specific sample rate f, see the green lines in the middle of Fig. 3. In radio-based locating systems the sampling rate is chosen to match the limitations of signal processing that inherit from the available bandwidth of the system. This also limits the accuracy of the resulting positions by design. For instance, the system we use for our experiments uses a sampling frequency of $f = 101.875\text{MHz}$, which approximately results in a distance of $c/f = c/101.875\text{MHz} \approx 2.94\text{m}$ (with c the speed of light) between two sampling points. However, an over-determined system and clever interpolation between samples results in a lateral position error that is much smaller.

As the resolution of the timing offsets Δt^i is often more fine-grained than integral multiples of $1/f$ (see the green lines that represent the sampling points among the CIR windows in the middle of Fig. 3), we have to re-adjust the discrete CIRs into the same timing units. Hence, we (1) up-sample the signal by a resampling factor n (see the red lines in Fig. 3), (2) shift it sample-wise by the integral timing offsets $\Delta t^i/f \cdot n$, and (3) down-sample the signals by its resampling factor n again. We resample the CIR by a factor of $n = 100$ (resulting in a distance of 2.94cm between two samples) and use an anti-aliasing FIR low-pass filter with kaiser window that also compensates the delay introduced by the filter.

Now we have the CIRs calibrated and sampled at common timing units. We further align the CIRs inside an $N \times M$ matrix with N antennas and $M > \text{len}(CIR)$ samples, that each CIR has the correct relative timing to each other and the median of all center of columns m is $M/2$. The median will compensate a small number of wrong detections of CIR, which can be caused by falsely detecting the CIR window start position. Samples not occupied by CIR samples will be padded by zeros. We further interpret the matrix $N \times M$ as an image with its real and imaginary parts as different channels.

B. Normalization of Data

Usually, ML (and DL in particular) requires a normalization of the dataset. In the training of an ML-model statistical properties (e.g. as mean and a standard deviation) are only computed from the training data, not from the validation or test data. Even if statistical properties are not explicitly computed the model converges in respect to the statistical properties of the underlying training data set. Hence, we must standardize the statistics for the validation and test data with the ones computed from the training data.

[1]We assume that we can install reference transmitters at positions with low multicast profiles. We then derive the offsets by reverse ToA-estimation.

In deep learning we often *center* the data and hence remove statistical dependencies between training and test data. Normalization techniques are highly application- and data-dependent. In image classification we subtract the mean image of the training data from a test sample. We then see if the model captures statistical dependencies of the underlying data. Often we can think of it as a way to normalize the dataset such that it has a zero mean and standard deviation of 1.

Initial trials showed poor results for conventional normalization methods. The main reason is that the correlation signals are highly affected by non-linear effects. Hence, we do not subtract a *mean* correlation from the data but we normalize each input on its own. We consider the signals from n antennas units as a single set. We use a width of m=60 to describe a correlation over time in its real and imaginary part, resulting in an $n \times m \times 2$ matrix. For each set we take the minimum and subtract that from each value. We next divide by the maximum value to effectively scale the values to $[0; 1]$. We apply this to real and imaginary signals separately.

Afterwards we combine the real and imaginary signals with their corresponding correlations values into a 2-channel correlation image of shape $120 \times 12 \times 2$ (that also considers recalibration and padding) as shown in Fig. 3 (right).

V. Experimental Setup

To validate our DL approach we recorded several training data sets using different setups. We describe our measurement infrastructure in Sec. V-A, our datasets in Sec. V-B, and our deep learning setup and model configuration in Sec. V-C.

A. Measurement Infrastructure

The core of deep learning methods is a large dataset that is used to train, validate and test the model. In addition to the CIR data, for the training and the evaluation of our model we also need precise ground truth reference positional data to label our training and test data sets. We obtain such labels with a Nikon iGPS system, i.e., an optical laser-based tracking system with a mean average error both vertically and horizontally below $1mm$ and an update rate of $30Hz$.

For our experiments we also need a radio-based locating system that delivers a stream of channel impulse responses. We generated CIR data with a custom radio-based locating system that runs in the globally license-free ISM (industrial, scientific, and medical) band of 2.4 GHz and that uses around 80 MHz bandwidth [39]. Miniaturized transmitters use the available bandwidth to generate short broadband signal bursts together with identification sequences on which we correlate on the antenna units. Fig. 4 illustrates the signal processing chain. The system distinguishes fixed reference transmitters for calibration purposes from M moving transmitters. All transmitters emit tracking burst signals, which are received by N receiving antennas. Our installation uses 12 antennas that receive signals from up to 144 different transmitters. Mobile tags emit up to 2,000 tracking bursts per second (we use transmitters with 200 positions per second). The locating system allows to receive 50,000 of those signal bursts per

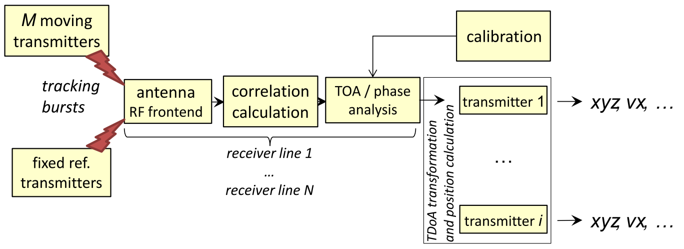


Fig. 4. RedFIR's signal processing chain.

second (per antenna). For each of the 12 receiver lines FPGAs correlate the burst sequences to obtain the correlation, i.e., the channel impulse response. We ignore the ToA analysis and work directly on the CIR streams. As the receivers are synchronized all the CIRs have a common time base.

We generate the data in the Fraunhofer IIS L.I.N.K. (localization, identification, navigation, communication) test center that provides a unique test ground on 1,400m^2, see Fig. 5. We used the following platforms to collect our training data in order to capture different properties of tracking.

Positioning System. We use a crane-like apparatus that approaches any 3D position in the area with repeating accuracy of $<2mm$ at a maximum speed of $3m/s$. Fig. 5 shows the crane as it passes through a construction of absorber walls. We use the positioning system to capture a homogeneously distributed dataset that covers a larger area at a fixed height. The positioning system provides 25 position per second which, however, is not as accurate as the iGPS position. Hence, we mounted not only the RF-tags but also the iGPS transmitters to the positioning system. As the system runs slowly we simply interpolate the reference positions (as the update rate of iGPS is lower than that of RedFIR).

Mobile Robot. We use a Segway RMP-210 with a maximum speed of $30km/h$ and an acceleration of $2m/s^2$ to capture highly dynamic tracking data. As a reference we take the iGPS position and interpolate intermediate reference positions by interpolation and odometry information.

Human. We also use a body-mounted apparatus that captures movements of persons. In general that allows to capture higher speeds, but usually the speed is below 10 km/h. We use several iGPS transmitters to determine the position of a mobile RF-tag that is located near the person's neck [40].

B. Datasets

Fig. 6 shows the measurement trajectories of our datasets. The platforms follow the trajectory and we record 200 correlation signals per antenna and second. As the iGPS system only delivers $30Hz$ ground truth positions we interpolate both the ground truth positions and timestamps, see Sec. V-A.

TABLE I
DESCRIPTION OF DATASETS.

Dataset	# Samples	Covered Area (w×h)	Height	Platform
Meander	200,390 (211,416)	$13m \times 20m$	2.5m	Pos.-Sys.
Zig-Zag	304,120 (349,025)	$22m \times 19m$	0.29m	Segway
Random Walk	404,687 (691,680)	$45m \times 30m$	0.96m - 2.1m	Human
Displaced Rectangles	92,724 (218,752)	$5m \times 14m$	2.8m	Pos.-Sys.

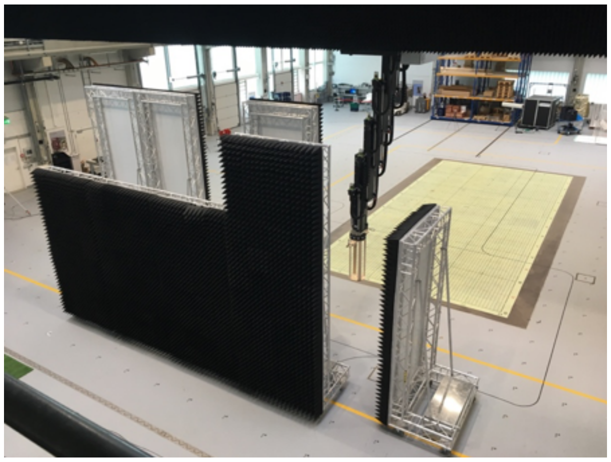


Fig. 5. Fraunhofer L.I.N.K. test center and positioning system.

Table I specifies our datasets. From the total number of recorded training samples we only select mostly complete sets (whenever we receive correlations from 11 of the 12 antennas). However, in a few cases correlations are corrupt, or there are warnings due to a bad signal-to-noise ratio. We then discard such measurements. However, due to the heavy multipath in the displaced rectangles dataset (the rectangles used the path from Fig. 5) the system is only able to correlate on 92,724 out of 218,752 burst signals to decode the CIRs. While we recorded 3D positions in any cases we fixed the z-coordinate where possible as we only evaluate for the 2D positional accuracy (as the sub-optimal geometry of the RF-antennas adds a bias to our evaluation). However, we could not fix the height of the transmitters on the random walk.

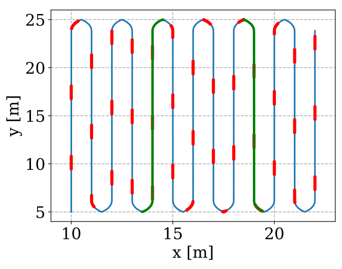
C. Deep Learning Setup

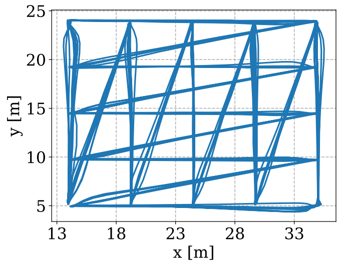
All our experiments run on a desktop machine equipped with an Intel Xeon E5-1620 v4 CPU@3.5GHz (4 cores, 8 threads, 10MB cache), 16GB of main memory, and an Nvidia GeForce GTX1080 GPU with 32GB memory. We implemented all our algorithms in C++14 on Ubuntu 16.04 LTS and used the CAFFE deep learning framework.

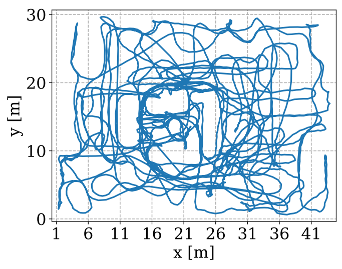
In pre-tests we evaluated several different and well-known deep learning architectures such as AlexNet, VGG-16, VGG-19, and GoogLeNet [41]. It turned out that the GoogLeNet not only offers the best trade-off between depth of the network and number of parameters (and hence the training time) but also benefits from its inception modules. An inception module is a 1×1 convolution that reduces the dimensionality of a feature map. These are used prior to computationally intense 5×5 and 3×3 convolutions. The GoogLeNet has 22 layer, uses 9 inception modules, and has 2 intermediate classifiers.

We made the following changes to the GoogLeNet architecture in order to facilitate the size of our correlation input:

- We replaced each of the 3 softmax classifiers by affine regressors (Euclidean distance).


(a) Meander.


(b) Zig-Zag.


(c) Random Walk.

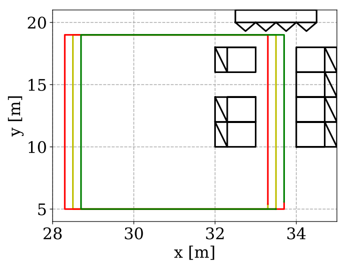

(d) Displaced Rectangles.

Fig. 6. Datasets we use for our experiments: Meander recorded with the positioning system; Zig-Zag recorded using the Segway RMP-210; Random Walk recorded using a human that builds up a construction in the test center; Displaced Rectangles recorded with the positioning system, see also Fig. 5.

- We replaced the fully connected (FC) layer that has 1,000 output units (before the classifier) by an FC layer of 2 units outputting a vector of positions (x, y).
- We modified the max-pooling layer after the 2nd inception module to have a kernel size of 2 instead of 3, the avg-pooling layer at the 1st and 2nd classifier to a kernel size of 3 instead of 5, and the max-pooling layer before final inception modules to a kernel size of 2 instead of 3.

For training we apply stochastic gradient descent (SGD) with a starting learning rate of $5 \cdot 10^{-4}$ and an inverse decay, and applied a batch size of 50 for training and 10 for testing.

VI. Results

To get a baseline positioning performance of the RF-system we extract the ToAs under LoS conditions using the inclination point method and run a Levenberg-Marquardt optimizer to obtain positions. For each set of ToAs we run the optimization 12 times (with each ToA once being set to 0 for TDoA estimation) and select the iteration with the lowest error term (best fit of TDoAs to position). On the Zig-Zag dataset we obtain an MAE of 0.804m, a CEP of 0.316m, and a CE95 of 1.49m if we skip extreme outliers that are out of range. (In practice phase analysis and motion models improve accuracy.)

A. General Performance Evaluation

For a simple performance experiment, we divide the datasets into training (80%) and testing (20%) set. We uniformly sample among the data points. Later, we evaluate the model by the whole set and employ the euclidean distance as a quality measure for the accuracy of the model.

Table II shows the results for our datasets. We achieve the best results w.r.t. all metrics on the *Displaced Rectangles* (Fig. 6 (d)) and *Meander* (Fig. 6(a)) datasets. However, the horizontal distance between the rectangles in *Displaced Rectangle* is only $0.2m$. But anyway, together with the multipath propagation (which enriches the information in the CIR) the model manages to estimate the position exceptionally good. The ZigZag (Fig. 6 (b)) gives us the hint that the speed of the Segway system has an effect on the CIRs. The performance is worst for the Random Walk 6(c) dataset because the height of the mobile tag varies. The model did not see any information of the z-axis and hence cannot generalize the measurement on the xy-plane. With a CEP that is smaller than the MAE

TABLE II
RESULTS FOR GENERAL PERFORMANCE.

Dataset	CEP	CE95	MAE
Meander	$15.6cm$	$36.4cm$	$17.4cm$
Zig-Zag	$24.1m$	$67.3m$	$29.1cm$
Random Walk	$30.3cm$	$86.8cm$	$36.2cm$
Displaced Rectangles	$10.2cm$	$24.3cm$	$11.6cm$

and a high CE95 we can easily filter out the outliers in a post-processing step. But on all our datasets (even on the Zig-Zag with high velocity and the LoS datasets) our DL-approach considerably outperforms the Levenberg-Marquardt optimization on the extracted ToAs.

B. Slicing Evaluation

Randomly sampling training and testing data naively does not elaborate whether the model generalized over the training data or overfitted the training data. To check for generalization we construct two additional scenarios on the Meander dataset:

1) Short-Slice (SS): We use the correlations from the red slices in Fig. 6(a) to test the model, while we use the rest to train the model. The test slices are approximately $1m$ long. This setup evaluates small-scale generalization [42].

2) Long-Slice (LS): We use the correlations from the green lines in Fig. 6(a) to test the model while we use the rest for training. This evaluates large-scale generalization.

Fig. 7 shows color-coded error plots for SS (left) and LS (right). SS has an MAE of $26.5cm$, a CEP of $23.6cm$ and CE95 of $57.1cm$, while LS has an MAE of $33.9cm$, a CEP of $26.5cm$ and a CE95 of $77.1cm$.

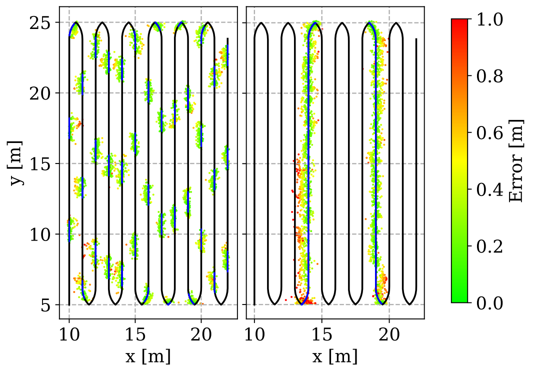


Fig. 7. Evaluation of the SS and LS

Both test results are worse than the results that we achieve using the naive sampling approach (which however, did not check for generalization). But both results also show that our model generalizes over the training data set. Hence, without having seen the test data points in the training data set the model produces viable position estimates that are even better than the baseline optimization.

C. Architecture Evaluation

We modified popular architectures, i.e., AlexNet, GoogleLeNet, VGG-16 and VGG-19, according to Sec. V-C and applied our correlation data to them. We further modified the GoogleNet, see Tab. III. The *-Re* modification preserves the CIRs further down the network (we modified the initial convolutional and pooling layers). The *Re-NoP* preserves the size of correlation image (by removing the pooling layer between the first convolutional and normalization layer). We also defined *Small-Net* as a cut-off from the GoogLeNet architecture (we removed the inception layers between the root and the first intermediate output).

Fig. 8 shows the CDFs of the network architecture trained and tested according to the LS scheme. The graph also shows the CDF of the Meander dataset according to Sec. VI-A with gray dotted line with the GoogLeNet and its modification G-Re-NoP as baseline for comparison. Table III specifies the evaluated architecture parameters together with inference time per 1000 samples, MAE and CEP. GoogLeNet and VGG-19 are on par especially on the CE95 level. Surprisingly, VGG-16 outperforms VGG-19 with its modifications. AlexNet has the worst overall performance compared to the other network architectures (as it is comparably shallow but mostly fully connected throughout the network).

The G-Re-Nop provides the best shaped CDF and MAE, but its inference time is considerably higher than that of others and also not feasible for practical applications of RLTS. Comparing GoogLeNet and Small-Net, we can see that they are on par w.r.t. MAE and CEP. However, Small-Net has nearly 4 times less parameters and a 6 times faster inference time. The modified Small-Net-Re has a slightly better CDF than the Small-Net, but with the cost of a larger number of parameters (approx. 2 million vs. 11 million) which also results in higher inference times.

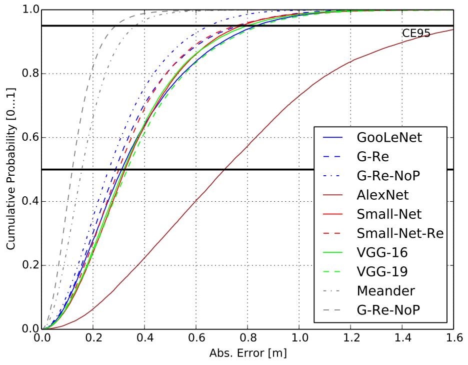


Fig. 8. Cumulative Probability over the Meander dataset.

TABLE III
RESULTS AND PARAMETERS OF DIFFERENT ARCHITECTURES.

Network	**# Params**	**Avg. FP** (ms)	**MAE** (cm)	**CEP**
GoogLeNet	6,894,976	66.30	0.36m	0.31m
G-Re	7,422,336	130.68	0.33m	0.28m
G-Re-NoP	8,778,112	411.68	0.29m	0.26m
AlexNet	34,535,104	24.46	0.79m	0.71m
Small-Net	2,113,664	10.83	0.36m	0.32m
Small-Net-Re	11,938,944	37.70	0.34m	0.30m
VGG-16	39,883,904	158.24	0.36m	0.32m
VGG-19	45,192,320	197.18	0.38m	0.33m

D. Multipath Scenario

In reality there are a number of situations where the signal is attenuated or deteriorated. Real world environments often include obstacles that facilitate multipath propagation. To see if our approach also manages to mitigate the effect of multipath we recorded the *Displaced Rectangles* dataset.

The rectangles in Fig. 6 (d) illustrate the trajectory. We use two of the three (red/left, green/right) rectangles for training and the middle one (yellow) for testing. Fig. 5 shows how we placed absorber walls on the right side of the dataset. The perpendicular part on the rectangles' right side heavily suffers under multipath propagation. We ended up with a dataset of 90K CIR inputs and divided them such that the training set consisted of 60K and testing set 30K correlations.

Fig. 9 left shows color-coded result of state-of-the-art (SoTA) extended Kalman filter using a constant acceleration motion model that uses ToAs and phase information as input. Fig. 9 right shows the results of our method. We use the yellow/middle trajectory for testing and a median filter for post-processing. We see that classic ToA-estimation and Kalman post-processing heavily suffers from NLoS situations. While ToA together with the transition matrices of the filter perform very well on the left side (MAE 14.8cm, CEP 14.1cm, CE95 27.0cm in gray rectangle left) the highly non-linear effects on the right side cannot be resolved (MAE 2,11m, CEP 1.45m, CE95 5.29m in gray rectangle right). Our approach is slightly worse in the LoS area with an MAE of 15.4cm, a CEP of 14.6cm and a CE95 of 28.3cm in the left rectangle. But most impressive is the NLoS performance: with a CEP of 22.9cm (MAE: 29.2cm, CE95: 68.3cm) our approach computes accurate positions even under heavy multipath (overall MAE: 17.3cm, CEP: 13.7cm, CE95: 45.0cm). This shows that our approach successfully resolves multipath propagation.

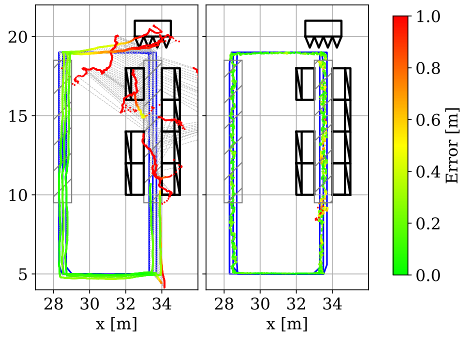


Fig. 9. Results with multipath scenario

VII. Conclusion

This paper presents a position estimation based on deep learning methods that directly works on the channel impulse responses of TDoA-based locating systems. We provided details of our signal and data preprocessing and show the efficiency of our approach in different real world setups. While our approach keeps up with conventional signal processing approaches under line-of-sight conditions it outperforms previous approaches under heavy multipath propagation.

References

[1] T. Nowak and A. Eidloth, "Dynamic multipath mitigation applying unscented kalman filters in local positioning systems," *Intl. J. Microwave and Wireless Technologies*, vol. 3, pp. 365–372, 2011.

[2] J. He, Y. Geng, F. Liu, and C. Xu, "Cc-kf: Enhanced toa performance in multipath and nlos indoor extreme environment," *IEEE Sensors J.*, vol. 14, no. 11, pp. 3766–3774, 2014.

[3] R. Exel and T. Bigler, "Toa ranging using subsample peak estimation and equalizer-based multipath reduction," in *Proc. IEEE Wireless Communications and Netw. Conf.*, (Istanbul, Turkey), pp. 2964–2969, 2014.

[4] M. Driusso, F. Babich, F. Knutti, M. Sabathy, and C. Marshall, "Estimation and tracking of lte signals time of arrival in a mobile multipath environment," in *Proc. 9th Intl. Symp. Image and Signal Processing and Analysis*, (Zagreb, Croatia), pp. 276–281, 2015.

[5] S. Al-Jazzar, J. Caffery, and H. R. You, "A scattering model based approach to nlos mitigation in toa location systems," in *Proc. 55th IEEE Conf. Vehicular Technology*, (Birmingham, AL), pp. 861–865, 2002.

[6] S. Al-Jazzar and J. Caffery, "Ml and bayesian toa location estimators for nlos environments," in *Proc. 56th IEEE Conf. Vehicular Technology*, (Vancouver, BC), pp. 1178–1181, 2002.

[7] L. Li and J. L. Krolik, "Simultaneous target and multipath positioning," *IEEE J. Selected Topics in Sign. Proc.*, vol. 8, no. 1, pp. 153–165, 2014.

[8] J. He, Y. Geng, and K. Pahlavan, "Toward accurate human tracking: Modeling time-of-arrival for wireless wearable sensors in multipath environment," *IEEE Sensors J.*, vol. 14, no. 11, pp. 3996–4006, 2014.

[9] P. Meissner, E. Leitinger, and K. Witrisal, "Uwb for robust indoor tracking: Weighting of multipath components for efficient estimation," *IEEE Wireless Communications Letters*, vol. 3, no. 5, pp. 501–504, 2014.

[10] N. Garcia, H. Wymeersch, E. G. Larsson, A. M. Haimovich, and M. Coulon, "Direct localization for massive mimo," *IEEE Trans. Signal Processing*, vol. 65, no. 10, pp. 2475–2487, 2017.

[11] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *2015 Intl. Conf. Computer Vision*, (Santiago de Chile, Chile), pp. 2938–2946, 2015.

[12] D. Mascharka and E. Manley, "Lips: Learning based indoor positioning system using mobile phone-based sensors," in *Proc. 13th IEEE An. Consumer Communic. Netw. Conf.*, (Las Vegas, NV), pp. 968–971, 2016.

[13] A. Martinez Sala, R. Quir'os, and E. L'opez, "Using neural networks and active rfid for indoor location services," in *Proc. Europ. Workshop Smart Objects: Sys., Tech. and App.*, (Ciudad, Spain), pp. 1–9, 2010.

[14] J. Luo and H. Gao, "Deep belief networks for fingerprinting indoor localization using ultrawideband technology," *Int. J. Distrib. Sens. Netw.*, vol. 2016, no. 18, pp. 18:18–18:18, 2016.

[15] S. Y. M. Vaghefi and R. M. Vaghefi, "A novel multilayer network model for toa-based localization in wireless sensor networks," in *Proc. Intl. Jnt. Conf. Neural Networks*, (San Jose, CA), pp. 3079–3084, 2011.

[16] P. Singh and S. Agrawal, "Tdoa based node localization in wsn using neural networks," in *Proc. Intl. Conf. Communication Systems and Network Technologies*, (Gwalior, India), pp. 400–404, 2013.

[17] A. Lewandowski, V. Kster, C. Wietfeld, and S. Michaelis, "Support vector machines for non-linear radio fingerprint recognition in real-life industrial environments," in *Proc. ION Intl. Technical Meeting*, (San Diego, CA), pp. 628–634, 2011.

[18] C.-S. Chen, "Artificial neural network for location estimation in wireless communication systems," *Sensors*, vol. 12, pp. 2798–2817, 2012.

[19] G. Flix, M. Siller, and E. N. lvarez, "A fingerprinting indoor localization algorithm based deep learning," in *Proc. 8th Intl. Conf. Ubiquitous and Future Networks*, (Vienna, Austria), pp. 1006–1011, 2016.

[20] R. Kuo, W. Tseng, F. Tien, and W. Liao, "Application of an artificial immune system-based fuzzy neural network to a rfid-based positioning system," *J. Comp. Indust. Eng.*, vol. 63, no. 4, pp. 943–956, 2012.

[21] V. Savic and E. G. Larsson, "Fingerprinting-based positioning in distributed massive mimo systems," in *Proc. 82nd IEEE Conf. Vehicular Tech.*, (Boston, MA), pp. 1–5, 2015.

[22] Z. Iqbal, D. Luo, P. Henry, S. Kazemifar, T. Rozario, Y. Yan, K. Westover, W. Lu, D. Nguyen, T. Long, J. Wang, H. Choy, and S. Jiang, "Accurate real time localization tracking in a clinical environment using bluetooth low energy and deep learning," *arXiv/1711.08149*, 2017.

[23] L. Yu, M. Laaraiedh, S. Avrillon, and B. Uguen, "Fingerprinting localization based on neural networks and ultra-wideband signals," in *Proc. IEEE Intl. Symp. Signal Processing and Information Technology*, (Bilbao, Spain), pp. 184–189, 2011.

[24] W. Li, T. Zhang, and Q. Zhang, "Experimental researches on an uwb nlos identification method based on machine learning," in *Proc. 15th IEEE Intl. Conf. Comm. Tech.*, (Guilin, China), pp. 473–477, 2013.

[25] S. Marano, W. M. Gifford, H. Wymeersch, and M. Z. Win, "Nlos identification and mitigation for localization based on uwb experimental data," *IEEE J. Selected Areas in Communications*, vol. 28, no. 7, pp. 1026–1035, 2010.

[26] X. Cui, H. Zhang, and T. Gulliver, "Threshold selection for ultra-wideband toa estimation based on neural networks," *J. Networks*, vol. 7, no. 9, pp. 1311–1318, 2012.

[27] V. Savic, E. G. Larsson, J. Ferrer-Coll, and P. Stenumgaard, "Kernel methods for accurate uwb-based ranging with reduced complexity," *IEEE Trans. Wireless Communic.*, vol. 15, no. 3, pp. 1783–1793, 2016.

[28] S. Ergüt, R. Rao, O. Dural, and Z. Sahinoglu, "Localization via tdoa in a uwb sensor network using neural networks," in *Proc. Intl. Conf. Communications*, (Beijing, China), pp. 2398– 2403, 2008.

[29] Y. Jin, W. Soh, and W. Wong, "Indoor localization with channel impulse response based fingerprint and nonparametric regression," *IEEE Trans. Wireless Communications*, vol. 9, no. 3, pp. 1120–1127, 2010.

[30] N. Ghourchian, M. Allegue-Martinez, and D. Precup, "Real-time indoor localization in smart homes using semi-supervised learning," in *Proc. 31st. AAAI Conf. Art. Intel.*, (San Francisco, CA), pp. 4670–4677, 2017.

[31] X. Wang, L. Gao, S. Mao, and S. Pandey, "Csi-based fingerprinting for indoor localization: A deep learning approach," *IEEE Trans. Vehicular Technology*, vol. 66, no. 1, pp. 763–776, 2017.

[32] X. Wang, L. Gao, and S. Mao, "CSI phase fingerprinting for indoor localization with a deep learning approach," *IEEE Internet of Things J.*, vol. 3, no. 6, pp. 1113–1123, 2016.

[33] X. Wang, X. Wang, and S. Mao, "Cifi: Deep convolutional neural networks for indoor localization with 5 ghz wi-fi," in *Proc. IEEE Intl. Conf. Communications*, (Paris, France), pp. 1–6, 2017.

[34] X. Wang, L. Gao, and S. Mao, "Biloc: Bi-modal deep learning for indoor localization with commodity 5ghz wifi," *IEEE Access*, vol. 5, pp. 4209–4220, 2017.

[35] J. Tiemann, J. Pillmann, and C. Wietfeld, "Ultra-wideband antenna-induced error prediction using deep learning on channel response data," in *Proc. 85th Vehic. Techn. Conf.*, (Sydney, Australia), pp. 1–5, 2017.

[36] J. Vieira, E. Leitinger, M. Sarajlic, X. Li, and F. Tufvesson, "Deep convolutional neural networks for massive MIMO fingerprint-based positioning," (Montreal, Canada), pp. 1–6, 2017.

[37] M. Z. Comiter, M. B. Crouse, and K. H. T., "A structured deep neural network for data driven localization in high frequency wireless networks," vol. 9, pp. 21–39, 05 2017.

[38] C. Xiao, D. Yang, Z. Chen, and G. Tan, "3-d ble indoor localization based on denoising autoencoder," *IEEE Access*, vol. 5, pp. 12751–12760, 2017.

[39] C. Mutschler, H. Ziekow, and Z. Jerzak, "The debs 2013 grand challenge," in *Proc. 7th ACM Intl. Conf. Distributed Event-based Systems*, pp. 289–294, 2013.

[40] T. Feigl, C. Mutschler, and M. Philippsen, "Human Compensation Strategies for Orientation Drifts," in *Proc. 25th IEEE Intl. Conf. Virtual Reality and 3D User Interfaces*, (Reutlingen, Germany), 2018.

[41] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, (Boston, MA), pp. 1–9, 2014.

[42] C. Löffler, S. Riechel, J. Fischer, and C. Mutschler, "Evaluation criteria for inside-out indoor positioning systems based on machine learning," in *Proc. 9th Intl. Conf. Indoor Positioning and Indoor Navigation*, (Nantes, France), 2018.