# An Advanced Method for Pedestrian Dead Reckoning using BLSTM-RNNs

Marcus Edel* and Enrico Köppe**
*Freie Universität Berlin/Mathematics and Computer Science, Berlin, Germany. Email: marcus.edel@fu-berlin.de
**BAM Federal Institute for Material Research and Testing, Berlin, Germany. Email: enrico.koeppe@bam.de

***Abstract*—Location estimation and navigation, especially on smartphones has shown great progress in the past decade due to its low cost and ability to work without additional infrastructure. However, a challenge is the positioning, both in terms of step detection, step length approximation as well as heading estimation, which must be accurate and robust, even when the use of the device is varied in terms of placement or orientation. In this paper, we propose a scheme for retrieving relevant information to detect steps and to estimate the correct step length from raw inertial measurement unit (IMU) data. This approach uses Bidirectional Long Short-Term Memory Recurrent Neural Networks (BLSTM-RNNs). Designed to take contextual information into account, the network can process data gathered from different positions, resulting in a system, which is invariant with respect to transformation and distortions of the input patterns. An experimental evaluation on a dataset produced from 10 individuals demonstrates that this new approach achieves significant improvements over previous attempts and increase the current state-of-the-art results even in the presence of variations and degradations. We achieved a mean classification rate of 98.5% and a standard deviation of 0.70 for 10000 different test sequences and an average error of 1.45% regarding the step length. Thus is the best result on the task gathered in the experiments compared with competing techniques.**

***Index Terms*—Indoor positioning, inertial tracking, dead reckoning, deep learning, machine learning**

## I. Introduction

Localizing and tracking people has recently received substantial attention, as knowledge about the current position can help in different areas. Especially in emergency situations, for example, after an earthquake or during the firefighting, the knowledge about the location of people can greatly improve search and rescue missions. Consider an example where firefighters in a building are enclosed by smoke and fire. If a map of the building can be constructed while the firefighters are within the building, an automated system can re-route the people to the exit. In addition, the generated map of the environment, can be used to intelligently coordinate the actions of the rescue workers to effectively search the environment and at the same time reduce the time people are exposed to potential threats.

Pedestrian dead reckoning (PDR) is a simple technique. The displacement of the user from a starting point is given by the number of steps they have taken, the direction of each step that was taken and the length of each step that was taken. Typically, this is achieved by using the data of an IMU. However, there are two crucial problems to develop PDR systems with low-cost sensors which can be found especially in smartphones. One is to identify the correct number of steps taken without knowledge of the device placement. The other is to maintain a high-accuracy step length approximation even over a long time. For example, an error of 2% over a long trajectory can place the user in an incorrect room due to the error accumulation over time.

In this paper, we propose a mobile step detection and step length estimation system based on BLSTM-RNNs and linear regression to alleviate the aforementioned problems.

The remainder of this paper is organized as follows: Sec. II presents the related works for PDR systems focusing on step detection and step length approximation. Sec. III outlines the system architecture, motivating the need for each component. Sec. V presents the first component of the PDR system that accurately counts the number of steps taken, regardless of the device placement. The network architecture is described in Section V-A. Sec. VI describes the technique to estimate the step length. The experimental setup is described in Sec. VII and an experimental comparison with end-to-end competing techniques on the data set is given in Sec. VIII-A. At the end, we present our conclusion and outlook for future work.

## II. Related Works

The success of the step detection and the step length estimation depends on the classification method, and what kind of technique is used to approximate the length of a single step. Most of the current research use static algorithms such as Multi-Layer Perceptrons [1], K-Means [2] clustering and a combination of Dynamic Time Warping [3] and other handcrafted methods. One of the main problems with these approaches is that the recognition accuracy is largely determined by the ability of the designer to come up with an appropriate set of features. Unfortunately, this turns out to be a daunting task which, must be redone generally for each new device placement. Therefore, we adopt a dynamic classification algorithm, which is a kind of a recurrent neural network named BLSTM-RNNs. BLSTM-RNNs have recently been shown to perform better, in terms of stability and recognition performance, than other algorithms for recognizing time-series patterns in several domains [4], [5].

## III. System Architecture

A pedestrian dead reckoning (PDR) system consists of: a contextual post-processor, generally based on numerical approximations, which calibrates the motion data to handle transformations and distortions of the input data. A fusion algorithm to estimate the device motion based on the calibrated sensor readings. A motion detector, which extracts motion of interest. A step length estimator, which often based on the motion detector calculates the step length, and a heading estimator which estimates the device heading.

The block diagram of the system architecture used here is shown in Fig. 1.

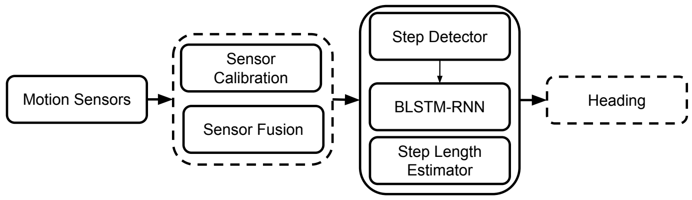

Fig. 1: System Architecture

Usually, the information carried from module to module is best represented as graphs with numerical information attached to the arcs. Typically each module is manually optimized out of its context. Then, the complete system is assembled, and a subset of the parameters of the modules is manually adjusted to maximize the overall performance. This last step is extremely tedious, time-consuming, and almost certainly suboptimal. A better alternative would be to somehow train the entire system so as to minimize the overall optimization process.

In this work, we focus on building robust techniques for the step detection and step length estimation module. Each component has been specifically designed to be used in various placements and orientations.

## IV. Calibration and Sensor Fusion

This comprises two main blocks, an automatic calibration algorithm, and an extended Kalman filter to combine data from the motion sensors. All modules are designed to be invariant to the rotation of the device over time. Achieved through coordinate transformation using quaternions as described in [6].

## V. Step Detector

The step detection system takes IMU data as input and returns a unit pulse when a step is detected, together with a classification of the step (e.g. forward, backward, running, walking). Existing work typically build more and more elaborate models to recognize classes of motion. Nevertheless, these approaches suffer from finding automatically the relevant parameters (e.g. signal processing) to deal with gesture variabilities. We instead developed hereafter a novel 3D step recognition method based on BLSTM-RNNs. Which has had great success in areas such as natural language processing [4], [7], but has yet to be considered for indoor tracking. Unlike directed graphical models like Hidden Markov Models, and other hand optimized step detectors the developed method works on raw sensor data. Furthermore, the recognition is position-invariant, meaning that only one learned sample is needed to recognize the same motion performed at other positions in the gestural space.

### *A. Long Short-Term Memory*

BLSTM-RNNs are dynamical systems introduced by S. Hochreiter and J.Schmidhuber [8]. They have been successfully applied to a number of tasks in computer vision, bioinformatics, and natural language processing. In all of these applications, given an input sequence $x = (x_1, ..., x_T)$, a standard recurrent neural network computes the sequences of hidden vectors $h = (h_1, ..., h_T)$ and output vectors $y = (y_1, ..., y_T)$ by recursively evaluation the following equations from time steps $t = 1$ to $t = T$:

$$h_t = f_{act}(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{1}$$

$$y_t = W_{hy}h_t + b_y \tag{2}$$

where $W$ denote the weight matrices, $b$ the bias vectors and $f_{act}$ the activation function of the hidden layer, often chosen to be the sigmoid or tanh function. However, standard RNNs tend to suffer from the vanishing gradient problem [9], thus limiting their access to long time lags. Therefore, the Long Short-Term Memory (LSTM) model [8] was developed to better find and exploit long-range context using special memory cells. Figure 2 illustrates a single LSTM memory block, which is the basic unit to model the network used in this paper. Furthermore, it is worthwhile to notice that a sequence often consists of both slow-moving and fast-moving components, of which only the former corresponds to long-term dependencies.

An LSTM layer consists of a number of recurrently connected memory blocks. Each block contains one or more recurrently connected memory cells and three multiplicative units, the input, output, and forget gates, which control the information flow inside the memory block. The surrounding network can only interact with the memory cells via the gates. In other words, these gates and the memory cell allow an LSTM unit to adaptively forget, memorize and expose the memory content. For the evaluation process described in this paper, we follow the implementation presented in [5].

### *B. Bidirectional LSTM*

One shortcoming of conventional RNNs is that they are only able to make use of previous context. In motion detection especially step detection, where fixed length is analysed at once, there is no reason not to exploit future context as well. Bidirectional RNNs (BRNNs) [10] do this by processing the data in both directions. They use two seperate hidden layers, which feed the data backwards to the same output layer. A BRNN computes the forward hidden sequence $\overrightarrow{h}$, the backward hidden sequence $\overleftarrow{h}$ and the output sequence $y$ by iterating the backward layer from $t = T$ to 1, the forward layer from $t = 1$ to $T$. At the end the output layer is updated. Combining BRNNs with LSTM gives bidirectional LSTM [4] which can access long-range context in both input directions. By resorting to bidirectional networks true online processing is impossible, due to the need of a complete data sequence to get an output. However, in case of the step detection it is

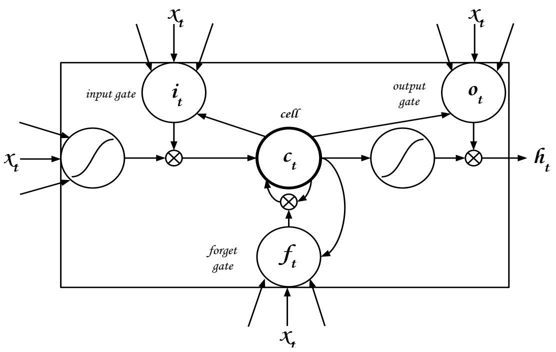

Fig. 2: LSTM memory cell with forget gate as described by Felix et al. in [11]. The self-connected node is the internal state. Nodes marked with $\otimes$ output the product of their inputs. Incoming arrows indicate recurrent edges and outgoing arrows have a fixed weight of 1.

sufficient to obtain an output at the end of a fixed motion sequence so that both passes, forward and backward, can be used fully during the detection process.

## VI. Step Length Estimation

The main idea of the step length estimation is to model the step length as a function of the sensor data, by fitting the data to a straight line. The basic idea has been empirically proven in [12].

### A. Step Length Functions

Due to the irregular nature of the pedestrian movements, the step length will change significantly depending on if the pedestrian is walking forward, walking backward, walking sideways or running forward, running backward, etc. It is, therefore, necessary to model separate step functions for the different motions, each with its own set of parameters:

$$\Lambda_{fw} = e + f\sqrt{Var(a_z)} \tag{3}$$

$$\Lambda_{fr} = g + h\sqrt{Var(a_z)} \tag{4}$$

$$\Lambda_{bw} = k + m\sqrt{Var(a_z)} \tag{5}$$

The coefficients $e, f, g, h, k$ and $m$ are the linear regression parameters to be estimated by the system. $Var(a_z)$, represents the variance of the z-axis acceleration during a step. So a motion classified as forward walk will be saved and used for the estimation of the forward walking step length function (3), a measurement classified as forward running for the forward running step length function (4), respectively a measurement classified as backward walking for the backward walking step length function, etc. For the classification of the different motions, we use the same network architecture as presented in Sec. V-A.

### B. Linear Regression

The idea is that the system can be reduced to a linear regression problem which provides a solution to the problem of finding the best fitting straight line through the set of samples. So the coefficient $e, f, g, h, k$ and $m$ are the linear regression parameters to be estimated by the system. In the standard formulation, a set of $N$ pairs of observations $\{Y_i, X_i\}$ is used to find a function relating the value of the dependent variable $Y$ to the values of an independent variable $X$. With one variable and a linear function, the prediction is given by the following equation:

$$\widehat{Y} = \alpha + \beta X s \tag{6}$$

This equation involves two free parameters which specify the intercept $\alpha$ and the slope $\beta$ of the regression line. The least square method defines the estimate of these parameters as the values which minimize the sum of the squares. Solving the normal equations gives the following least square estimates of $\alpha$ and $\beta$ as:

$$\alpha = \bar{Y} - \beta\bar{X} \tag{7}$$

with $\bar{Y}$ and $\bar{X}$ denoting the means of $X$ and $Y$ and

$$\beta = \frac{\sum_{i=1}^{N}(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{N}(X_i - \bar{X})^2} \tag{8}$$

The coefficient $\beta$ is equivalent to the parameters $f, h$ or $m$ in (3) - (5), depending on which function variable is used, and the coefficient $\alpha$ is equivalent to $a$ or $c$. Note that the coefficient $\alpha$ is set to zero for walking forward, forcing the step length function to go through the origin.
So, whenever a measurement is obtained, it is sorted into a fixed number of buckets based on the recognized motion. Note, that we use a low-pass filter to procedure a single averaged value for each bucket, which is used for the estimation of the regression coefficients. When a step is detected by the step detector V-A, an integration of the calibrated accelerometer data is started. After three steps, the integration stops and the obtained distance is divided by three to get the step length. At each step, the obtained function variable is stored, and the average of the three steps is used as a measurement. Whenever a new measurement is obtained, the regression coefficients for the corresponding state are re-estimated using least square fit as described above.

## VII. Experimental Setup

For all tests, ground truth data was obtained using a foot mounted IMU (3D-gyroscope, 3D-accelerometer, 3D-magnetometer). The ground truth data was obtained by placing labels along the route at 50 cm evenly spaced intervals and using a down facing hand-held camera to note the exact time when the foot touched the ground. For the evaluation and comparison with other competitive algorithms, we also used the data from a calibration IMU placed in the user's pocket, hand and day bag.

The input layer of the neural network consists in the concatenation of accelerometer and gyroscope information synchronized in time (six input values per time-step). Notice that our system relies only on the calibrated sensor data, without any additional feature extraction in opposition to most state-of-the-art methods. These data are linearly normalized between $-1$ and $+1$ according to the maximum value that sensors can provide. The forward and backward LSTM hidden layers are fully connected to the input layer and consist on five LSTM neurons each with full recurrent connections. The output layer has a size equal to the number of different step functions (forward, backward, sideways respectively for running and walking). The SoftMax activation function was used for the output-layer to give network responses between 0 and 1 at every time-step. Classically, these outputs can be considered as posterior probabilities of the input sequence to belong to a specific motion class at a given time-step.

Following Zaremba et al. [13], we used the mixed curriculum strategy to model our network.

To validate our approach, we recorded 10 hours of data consisting of a large number of activities. The data has been collected across multiple days, by 10 volunteers in various indoor and outdoor locations. Each person performed a mixed walk where all motion types were represented; sideways right and left, forward, backward and respectively walking and running. In the end, we used Adam [14] to train the model. Each update was performed using a minibatch with 128 sequences and a sliding window of 500 samples to extract a single sequence. We used a learning rate of 0.001 and $\beta_1$ and $\beta_2$ [14] were both set to 0.01. We trained each model for 30 epochs, with early stopping based on the validation set performance to prevent over-fitting. At test time, we evaluated each model on multiple sets of test examples. Each test set contains 10000 different sequences which don't overlap with the training set. For evaluation of a new motion sequence, we use a majority voting rule over the outputs along the sequence to determine the final motion class or time at which the step has been taken.

To determine the impact of the device placement, we examine various device positions, including static positions where the device is regardless of placement not moving, and dynamic placements where the validation is performed e.g. during a phone call. Furthermore, we compared the developed algorithm once without an initial step length and used the calibrated accelerometer velocity as described in section VI. Additionally we performed the experiments with an initial step length extracted from the training set.

## VIII. Results and Analysis

In this section, we compare and contrast the performance of our approach with leading PDR approaches for step detection and counting. The competing approaches we compare against are:

**Spectral analysis (STFT and CWT):** Algorithms [15]–[18] of this category convert the acceleration signal to the frequency domain using different algorithms: short-term Fourier transform, wavelet transform. Afterward, the dominant peak of the signal in the frequency domain is identified as the step frequency and used to detect the number of steps taken.

**Peak detection (PD):** When the sensor is placed on the user's foot, steps can easily be detected by identifying the stance phases of the foot corresponding to zero velocity periods. However, this approach can't be applied when the sensor isn't on the foot. Since, generally in other positions there isn't a period of zero velocity. However, bio-mechanic studies show that the swinging of the arm is synchronized with the foot motion. Consequently, swing events can be detected and the step detection can be expressed as a peak detection problem. Generally, algorithms [19], [20] of this category are using a signal peak detector, which recognized the local maximum or minimum within a sliding window in combination with an algorithm based on adaptive thresholds. By combining the two modules, the steps taken can be approximated in various positions.

**Robust pedestrian dead reckoning (R-PDR):** This promising technique [21] has attracted a large amount of attention recently due to its robustness. The method used to count the steps, is based on the fundamentals of human bipedal motion. The authors are using an enhanced zero crossing detector which makes use of symmetric or asymmetric motion to count the steps.

**Feedforward neural network (FNN):** Algorithms [22], [23] of this category first calculate the acceleration magnitude signal from the accelerometer signals. Define step boundaries by using the positive-going zero crossings of a low-pass filtered version of the accelerometer signal. Next the algorithms extract features (integral of the acceleration, magnitude between footfalls, standard deviation, mean value, etc.) from the filtered signal to train a feed-forward network. The network is then optimized using a standard optimization technique to approximate the step length.

**Inverted pendulum model (M1-M3):** In its simplest form, human gait can be described by an inverted pendulum model. From this mechanical model, necessary relations between the forward displacement and various measurable step variables can be obtained to estimate the step length. Algorithms described by Diego Alvarez et al. [24] and Harvey Weinberg [12] use the relationship between the vertical and the forward displacement to estimate the step length. Given by the equation:

$$M = K \cdot 2\sqrt{2lh - h^2} \tag{9}$$

Where $l$ stands for the leg length, $h$ for the vertical displacement of the center of mass during one step and $K$ is a constant which has to be calibrated for each person individually, using experimental data.

### A. Step Counting

We measured the accuracy of each algorithm by comparing the estimated step count with the ground truth values. Using an error rate metric defined as:

$$error = \frac{estimated - ground\ truth}{ground\ truth} \tag{10}$$

We observe that all of the algorithms had a median error rate of less than 12% and were more inclined to undercount than overcount the number of steps taken. In Table I it is shown that the proposed BLSTM with an average accuracy of 98.5% (mean error of 1.48%) outperforms other competing architectures during the validation.

| | CWT | R-PDR | STFT | PD | BLSTM |
|---|---|---|---|---|---|
| **Pocket** | 7.2 ± 1.95 | 5.5 ± 1.31 | 9.5 ± 1.62 | 10.2 ± 0.96 | 1.2 ± 0.71 |
| **Foot** | 6.7 ± 2.01 | 4.8 ± 1.22 | 7.7 ± 1.61 | 8.7 ± 0.99 | 1.0 ± 0.51 |
| **Daybag** | 7.3 ± 2.21 | 6.3 ± 1.53 | 9.8 ± 1.86 | 8.5 ± 1.81 | 1.4 ± 0.60 |
| **Hand** | 8.1 ± 2.92 | 5.1 ± 1.74 | 13.2 ± 1.9 | 16.7 ± 1.84 | 2.3 ± 0.89 |
| **Mean** | 7.32 | 5.43 | 10.05 | 11.02 | 1.48 |

TABLE I: Step counting error (%), mean and standard deviation, for individual subjects and positions.

Furthermore, BLSTMs tends to make faster and more robust estimations of the steps taken then the algorithms we compared against. It is interesting to note that for the competing algorithms, they typically attain good performance for some placements, but for two or three placements the accuracy is particularly poor. In some cases, the error is as high as 17%. This behavior is especially observed when data from multiple speeds are used. In particular, the placement in which the device is placed in the hand of the experimenter shows typically the highest errors. As the motion is typically prone to noisy repetitive motions. Out of the competing algorithms, the R-PDR algorithm performs reasonably well across all tested placements, with a maximum error of 5.43%. The strength of our approach is that the results are stable regardless of the placement, in particular, the errors are consistent below 3%.

## B. Step Length Estimation

As described in VII two different procedures has been employed to validate the various methods. In the first one, we compare the algorithm without an initial step length and used the calibrated accelerometer data from the first steps. In the second experiment, we used the ground-truth data as the initial basis for the individual step length functions. In addition to the validation of the step detection algorithms, we plotted in Table II the validation of the step length estimation model against the competing techniques.

Methods M1 to M3 including FNN show a tendency to overestimate the step length, as results vary between 106% and 129% of the ground truth data. The highest error and variance was obtained using method M1. BLSTMs in combination with the proposed regression algorithm shows the best results (98.7-102.8%). The results also effectively demonstrate that our approach significantly improves the tracking of a users position in different device placements.

Similar results are obtained for all methods using initial data estimated from the training data. This has application for wearable devices because this indicated that initial parameter estimation is not crucial for a robust estimation of the step length. That results in the conclusion that a time-consuming pre-calibration of the function parameters isn't necessary.

| | M1 | M2 | M3 | FNN | BLSTM - Regression |
|---|---|---|---|---|---|
| **Pocket** | 113.3 ± 7.8 | 109.4 ± 3.4 | 110.1 ± 7.5 | 108.1 ± 3.9 | 98.7 ± 2.6 |
| **Foot** | 110.8 ± 5.2 | 106.3 ± 2.9 | 108.0 ± 4.2 | 103.2 ± 3.1 | 99.4 ± 1.3 |
| **Daybag** | 117.2 ± 8.4 | 113.7 ± 5.5 | 115.0 ± 8.9 | 105.7 ± 4.5 | 100.9 ± 2.8 |
| **Hand** | 129.3 ± 9.5 | 118.2 ± 6.7 | 118.4 ± 9.1 | 117.6 ± 9.1 | 106.8 ± 3.0 |
| **Mean** | 117.68 | 111.9 | 112.88 | 108.65 | 101.45 |

TABLE II: Walking distance estimation (%) compared to the real walking distance.

| | M1 | M2 | M3 | FNN | BLSTM - Regression |
|---|---|---|---|---|---|
| **Pocket** | 111.5 ± 7.1 | 107.7 ± 3.2 | 108.4 ± 6.8 | 107.1 ± 3.5 | 99.1 ± 2.2 |
| **Foot** | 109.2 ± 4.6 | 105.4 ± 2.4 | 106.7 ± 4.5 | 103.2 ± 3.1 | 99.6 ± 1.5 |
| **Daybag** | 115.5 ± 7.8 | 112.1 ± 5.2 | 113.8 ± 8.5 | 105.7 ± 4.5 | 99.7 ± 2.9 |
| **Hand** | 130.1 ± 7.4 | 116.8 ± 6.7 | 116.3 ± 8.7 | 117.6 ± 9.1 | 104.8 ± 3.2 |
| **Mean** | 116.58 | 110.5 | 111.30 | 108.65 | 100.80 |

TABLE III: Walking distance estimation (%) compared to the real walking distance, using initial parameters estimated from the training data.

## IX. Conclusion and Outlook

We have proposed a novel approach to robustly detect steps by exploiting the context-sensitive characteristics of bidirectional Long Short-Term Memory models as well as the idea of using BLSTM networks as a basis to approximate the step length. The results presented in this paper demonstrate that BLSTM models in combination with modeling each motion as a single independent function outperformed state-of-the-art PDR systems.

Furthermore the results demonstrated that recurrent nets as used here for the step detection and step length approximation are especially interesting because they seem able to overcome restrictions generally placed on data by traditional machine learning approaches used in this area. Basically with recurrent nets, the assumption of independence between consecutive examples is broken, the assumption of fixed-dimension input is broken, and yet RNN models perform competitively with or outperform the state of the art on as shown for the step detection and step length estimation.

Future research should focus on the imbalance problem caused by the distortions existing in the dataset as well as improving the sensor data calibration. We also intend to investigate the effect of feature selection more deeply, which might lead to better generalization performance. Furthermore, we plan to perform more in-depth experiments with BLSTM RNNs and combination of BLSTMs with convolutional neural networks and to combine all techniques to improve the overall system accuracy.

## Acknowledgment

The research presented in this publication was conducted while the first author was employed by the Federal Institute for Materials Research and Testing.

## References

[1] G. Niezen and G. Hancke, "Evaluating and optimising accelerometer-based gesture recognition techniques for mobile devices," in *AFRICON*, 2009.

[2] J.-K. Min, B. Choe, and S.-B. Cho, "A selective template matching algorithm for short and intuitive gesture ui of accelerometer-builtin mobile phones," 2010.

[3] J. Liu, Z. Wang, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "Uwave: accelerometer-based personalized gesture recognition and its applications," in *PerCom 2009*, 2009, pp. 1–9.

[4] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[5] A. Graves, A. Mohamed, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," *CoRR*, 2013.

[6] E. Koppe, D. Augustin, A. Liers, and J. Schiller, "Self-calibration-method for an inertial navigation system with three 3d sensors," 2014.

[7] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[9] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," in, Kremer and Kolen, Eds., IEEE Press, 2001.

[10] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *Trans. Sig. Proc.*, vol. 45, no. 11, pp. 2673–2681, 1997.

[11] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: continual prediction with LSTM.," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[12] H. Weinberg, "Using the adxl202 in pedometer and personal navigation applications," *Analog Devices AN-602 Application Note*, 2002.

[13] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *CoRR*, vol. abs/1409.2329, 2014.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[15] A. Brajdic and R. Harle, "Walk detection and step counting on unconstrained smartphones," in *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '13, Zurich, Switzerland: ACM, 2013, pp. 225–234.

[16] M. Vetterli and J. Kovačevic, *Wavelets and Subband Coding*. 1995.

[17] A. Brajdic and R. Harle, "Walk detection and step counting on unconstrained smartphones," ser. UbiComp '13, 2013, pp. 225–234.

[18] V. Renaudin, M. Susi, and G. Lachapelle, "Step length estimation using handheld inertial sensors," *Sensors*, vol. 12, no. 7, p. 8507, 2012.

[19] D. H. H. J. W. Kim H. J. Jang and C. Park, "A step stride and heading determination for the pedestrian navigation system," *Journal of Global Positioning Systems*, pages, 2004.

[20] M. Susi, V. Renaudin, and G. Lachapelle, "Motion mode recognition and step detection algorithms for mobile phone users," *Sensors*, vol. 13, no. 2, p. 1539, 2013.

[21] Z. Xiao, H. Wen, A. Markham, and N. Trigoni, "Lightweight map matching for indoor localisation using conditional random fields," in *IPSN'14*, pp. 131–142.

[22] S. Y. Cho and C. G. Park, "MEMS Based Pedestrian Navigation System," in *The Journal of Navigation*, Seoul: The Royal Institute of Navigation, 2006, pp. 135–163.

[23] S. Beauregard, Haas, and Wpnc, "Pedestrian dead reckoning: a basis for personal positioning," *In Proc. 3rd Workshop Pos. Nav. Commun. (WPNC'06)*, pp. 27–36, 2006.

[24] D. Alvarez, R. Gonzalez, A. Lopez, and J. Alvarez, "Comparison of step length estimators from weareable accelerometer devices," in *EMBS 2006*, pp. 5964–5967.