Application of XGBoost Algorithm in Fingerprinting Localisation Task

Marcin Luckner(✉), Bartosz Topolski, and Magdalena Mazurek

Faculty of Mathematics and Information Science, Warsaw University of Technology, ul. Koszykowa 75, 00–662 Warszawa, Poland
mluckner@mini.pw.edu.pl, {topolskib,mazurekm}@student.mini.pw.edu.pl

Abstract. An Indoor Positioning System (IPS) issues regression and classification challenges in form of an horizontal localisation and a floor detection. We propose to apply the XGBoost algorithm for both tasks. The algorithm uses vectors of Received Signal Strengths from Wi–Fi access points to map the obtained fingerprints into horizontal coordinates and a current floor number. The original application schema for the algorithm to create IPS was proposed. The algorithm was tested using real data from an academic building. The testing data were split into two datasets. The first data set contains signals from all observed access points. The second dataset consist of signals from the academic network infrastructure. The second dataset was created to eliminate temporary hotspots and to improve a stability of the positioning system. The tested algorithm got similar results as reference methods on the wider set of access points. On the limited set the algorithm obtained the best results.

1 Introduction

Creation of an effective Indoor Positioning System (IPS) is an open issue. Commonly known and usage methods based on the Global Positioning System (GPS) are great outdoor but fail inside the buildings.

A fingerprinting based on Wi–Fi signals is a popular method that replaces GPS localisation. In this method, vectors of Received Signal Strengths (RSS) are mapped into horizontal coordinates by a regression method.

However, expectations for the localisation system are greater in the case of the IPSs. The acceptable accuracy of an outdoor localisation system is measured in dozens of meters when an expected error of IPS should not be greater than few meters.

Moreover, the indoor localisation raises floor detection issue that does not exist in an outdoor localisation task. An RSS vector should be mapped into a current floor by a classification method. It is extremely important to obtain a high accuracy in this classification task. A localisation with minimal horizontal error is useless if an object is localised on the wrong floor.

The research is supported by the National Centre for Research and Development, grant No PBS2/B3/24/2014, application No 208921.

K. Saeed et al. (Eds.): CISIM 2017, LNCS 10244, pp. 661–671, 2017.
DOI: 10.1007/978-3-319-59105-6_57

Therefore, we are still looking for new classification and regression methods that can be applied in fingerprinting localisation task.

In this paper, we applied the XGBoost algorithm [2] in fingerprinting localisation task. The XGBoost algorithm is a scalable tree boosting system that can be used both for classification and regression tasks. According to our knowledge, the algorithm was not applied in fingerprinting localisation task before. The original schema of the application was proposed.

The algorithm was tested on real data collected in an academic building. The localisation results were compared with a reference method. As a reference methods kNN and random forest were used.

The algorithm was tested on two data sources. The first source contains signals from all access points observed during the data collection. However, it was stressed [5] that such data may contain temporary access points. If such access points are present only in the learning set the accuracy obtained on the testing set will be decreased. Therefore, the second data source was limited to well–known access points that create the academic infrastructure.

The rest of the paper is structured as follows. Section 2 contains information on a fingerprinting method and measures of errors. Section 3 presents related work. Section 4 describes the XGBoost algorithm and its application for IPS. Section 5 presents the obtained localisation results. Finally, Sect. 6 presents brief conclusions.

2 Preliminaries

In this section we briefly describe a general fingerprinting method and the measures of an error used in this paper.

The fingerprinting approach to indoor localisation requires a model based on a dataset of measured Received Signal Strengths (RSS), which is later used to predict user's location on the base of the current RSS measurements. Our analysis was based on the data collected from a six-floor academic building. The building had an irregular shape and its outer dimensions were around 50 by 70 m and its height was 24 m.

The signals were measured using the following models of mobile phones: LG Nexus 4, Sony Xperia Z, and HTC One mini. All phones were working with Android 4.2 Jelly Bean and dedicated application for data collection [11].

The data were collected into the two independent series, used later as the training and the testing subset. Each subset consists of the vectors of the signal strengths from $\mathbb{R}^{570}$ labelled by the position of a point in which those signals were measured. The position information contains x and y - the horizontal coordinates, and f - the floor number. The training set contains the measurements from 1401 points gathered in a 1.5×1.5 m grid, and the test set contains the measurements from 1461 points gathered on another day in the grid shifted by 0.75 m in each horizontal direction. There were 40 measurements for each point, giving us 56020 training fingerprints and 58440 test fingerprints.

The quality of models was evaluated by two measures: a horizontal median error (**HME**) and an accuracy of the floor detection (**ACC**). The horizontal error for fingerprint $m \in \mathbb{R}^{570}$ was defined as

$$he(m) = \sqrt{(x_m - h_x(m))^2 + (y_m - h_y(m))^2)} \tag{1}$$

where x_m and y_m were the horizontal coordinates of the point where the fingerprint m was taken, and $h_x(m)$ and $h_y(m)$ were coordinates predicted for the fingerprint m. The floor error for fingerprint $m \in \mathbb{R}^{570}$ was defined as

$$fe(m) = |f_m - h_f(m)| \tag{2}$$

where f_m was the floor number where the fingerprint m was taken, and $h_f(m)$ was a floor number predicted by the model.

We defined a horizontal median error for the test set $\mathcal{T}$ as

$$\mathbf{HME} = \frac{\sum_{m \in \mathcal{T}} he(m)}{|\mathcal{T}|} \tag{3}$$

and an accuracy of the floor detection as

$$\mathbf{ACC} = \frac{\sum_{m \in \mathcal{T}} (1 - \mathrm{sgn}(fe(m)))}{|\mathcal{T}|} \tag{4}$$

3 Related Work

The comparison of indoor positioning system can be found in [3,4].

In [3] the authors presented an overview of the current trends in this localisation using a Wireless Sensor Network (WSN) with an introduction of the mathematical tools used to determine position.

Work [4] compared different methods of creating IPS model for a given radio map. The authors defined two categories: deterministic and probabilistic. The deterministic methods are based on distances between RSS vectors while probabilistic ones use the estimation of the probability that we are in a given localisation while a particular RSS vector was measured.

There were several works that discussed a usage of machine learning methods to create IPS for the same building but not necessary on the same data as used in this paper.

In work [10], the Particle Swarm Optimisation (PSO) algorithm was applied to the training process of a Multilayer Perceptron (MLP). In work [8] the k Nearest Neighbours (kNN) algorithm was used to estimate a current floor. In work [12] a localisation method for moving terminal was implemented with the usage of the Particle Filter method.

Finally, in works [5,7] the authors used the random forest. The authors built the independent models for the estimation of each coordinate (x, y, f).

The nature of the given problem justifies the usage of a random tree-based method [1]. The interference in a signal strength caused by walls and irregular

access point locations disallows any assumptions about the distribution of signal strengths. Moreover, those characteristics also imply highly non-linear interactions between variables, which are very well-handled by tree-based methods. Having this in mind we decided to use XGBoost algorithm and check if it can improve the results obtained with the random forest.

4 XGBoost Algorithm

In this section we show basics of tree boosting methods, and we discuss XGBoost algorithm, a scalable machine learning system for tree boosting.

The main difference between Random Forest (RF) and Gradient Boosted Machines (GBM) is that while in RF trees are built independent of each other, GBM adds a new tree to complement already built ones.

Assume we have data $\mathcal{D} = \{(x_i, y_i) : i = 1 \dots n,\ x_i \in \mathbb{R}^m,\ y_i \in \mathbb{R}\}$, so we have n observations with m features each and with a corresponding variable y. Let us define $\hat{y}_i$ as a result given by an ensemble represented by the generalised model

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i) \tag{5}$$

In our case f_k is a regression tree, and $f_k(x_i)$ represents the score given by the k-th tree to the i-th observation in data. We want to minimise the following regularised objective function in order to choose functions f_k.

$$\mathcal{L}(\phi) = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k) \tag{6}$$

l is the loss function. To prevent too large complexity of the model the penalty term Ω is included as follows:

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \|w\|^2, \tag{7}$$

where *gamma* and *lambda* are parameters controlling penalty for, respectively, the number of leaves T and magnitude of leaf weights w.

This penalty term is the first feature that is unique to XGBoost in comparison to general tree boosting methods. Its purpose is to prevent over-fitting and to simplify models produced by this algorithm.

In order to minimise the objective function an iterative method is used. In j-th iteration we want to add f_j, which minimises the following objective function:

$$\mathcal{L}^j = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(j-1)} + f_j(x_i)) + \Omega(f_j). \tag{8}$$

Using the Taylor expansion we can simplify this function, and derive a formula for loss reduction after the tree split from given node:

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{\left(\sum_{i\in\mathcal{I}_L} g_i\right)^2}{\sum_{i\in\mathcal{I}_L} h_i + \lambda} + \frac{\left(\sum_{i\in\mathcal{I}_R} g_i\right)^2}{\sum_{i\in\mathcal{I}_R} h_i + \lambda} - \frac{\left(\sum_{i\in\mathcal{I}} g_i\right)^2}{\sum_{i\in\mathcal{I}} h_i + \lambda}\right] - \gamma, \tag{9}$$

where $\mathcal{I}$ is a subset of the observations available in the current node, and $\mathcal{I}_L$, $\mathcal{I}_R$ are subsets of the observations available in the left and right nodes after the split. The functions g_i and h_i are defined as follows: $g_i = \partial_{\hat{y_i}^{(j-1)}} l(y_i, \hat{y_i}^{(j-1)})$, $h_i = \partial^2_{\hat{y_i}^{(j-1)}} l(y_i, \hat{y_i}^{(j-1)})$. This formula is used for finding the best split at any given node. It depends only on the loss function (through the first and second-order gradients) and the regularisation parameter γ. It is easy to see that this algorithm can optimise any loss function given by the user, as long as he can provide the first and second-order gradients.

In addition to introducing the regularised loss function, XGBoost provides two additional (compared to general GBM) features to prevent over-fitting. First, the weights of each new tree can be scaled down by a given constant η. It reduces an impact of a single tree on the final score, and leaves more room for next trees to improve the model. The second feature is a column sampling. It works in a similar way as random forests – each tree is built using only a column-wise sample from the training dataset.

Besides those improvements in terms of the algorithm, XGBoost also performs better than other tree boosting methods. It supports an approximate split finding, which improves the process of the building trees and scales very well with the number of CPU cores.

For our purposes we used regression with least squares objective for the horizontal localisation task and a multi–class classification using a softmax objective function for the floor detection. Both objectives are implemented in the 'xgboost' package for the 'R' language provided by the creators of the algorithm.

4.1 Application for IPS

Our task was to estimate of the coordinates (x_m, y_m, f_m) of the point where the fingerprint m was taken by predictions $(h_x(m), h_y(m), h_f(m))$. It was done by an ensemble of XGBoost models.

A straightforward application uses three models to estimate each of the coordinates separately. We proposed the different process for the training models. Instead of using two models for estimation of the horizontal coordinates in the whole building, we build two models for each floor of the building. By making each model more specific than the generic one we can make more accurate predictions on each floor. However, we are aware that this approach can lead to horizontal location errors caused by wrong floor prediction.

The application of XGBoost models for IPS was done using the following schema. The first model ϕ^f estimated the current floor

$$h_f(m) = \phi^f(m). \tag{10}$$

For each floor separate models were created for estimation of the horizontal coordinates. The coordinates x_m and y_m were estimated by the models ϕ^x_i and ϕ^y_i respectively, where $i \in [\min(f_m), \ldots, \max(f_m)]$.

According to recognised floor the x_m coordinate was estimated as

$$h_x(m) = \phi^x_{h_f(m)}(m). \tag{11}$$

Similarly, the y_m coordinate was estimated as

$$h_y(m) = \phi^y_{h_f(m)}(m). \tag{12}$$

5 Results

While the data provided measurements from 570 access points, work [5] concluded that it is not necessary to use all variables in the learning process. Therefore, we selected only 46 access points from the academic Wi–Fi network. The selection expelled all mobile and temporary access points (which was desirable from the practical point of view) but also made the models building and the predictions making much faster. However, for the sake of comparison we also tested the model based on all access points.

We compared the XGBoost model with random forests – using results from [5] – and with the kNN algorithm implemented for the need for the reference method in this work. It should be stressed that the model of the horizontal localisation in [5] was created for the whole building contrary to our model where the horizontal localisation was modelled for each floor separately.

5.1 Tests on Full Set of Access Points

In the first test all access points were taken into consideration. The number of trees in the model was fixed to 450 and the other parameters were adjusted to achieve the best performance on the learning set.

Table 1. Results of localisation on all 570 access points

Model	HME			ACC
	Median	Mean	80%	
XGBoost (2 models)	2.44	3.32	4.54	0.95
XGBoost (12 models)	2.34	3.37	4.42	0.95
Random forest	2.78	3.80	5.14	0.94
kNN	2.39	3.00	4.19	0.93

Table 1 compares the results obtained by the XGBoost algorithm, the kNN method, and the random forests. A quality of the algorithms is measured by the mean, median, and 80th percentile values of the horizontal error (**HME**) and by the accuracy (**ACC**).

The XGBoost algorithm was tested in two forms. The straightforward application used two models for the horizontal localisation and the application described in Sect. 4.1 used 12 models in order to do the same localisation.

We can notice that on the full dataset the kNN algorithm performs better than the XGBoost algorithm in terms of the horizontal error. We can also see

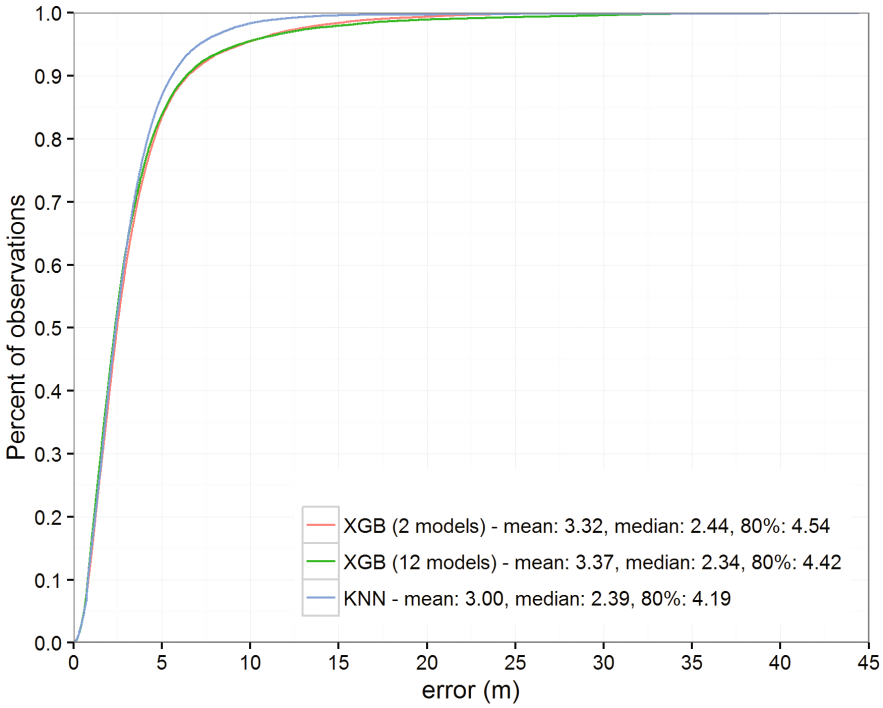

Fig. 1. Results of horizontal localisation on all access points

that using the XGBoost algorithm improves the horizontal median error over random forest by about 12%.

Figure 1 shows the distribution of the horizontal errors for the XGBoost and kNN models. The kNN algorithm reduced the number of errors from the range 4 to 20 m. Therefore, the differences between the means and gross errors are visible when the median error are nearly the same. The differences between two application schema of the XGBoost algorithm are ambiguous because one model has a lower median error when the other wins a comparison of the rest of the measures.

The XGBoost algorithm obtained the best accuracy in the floor detection task. Mostly, the classification error was not greater than one floor. However, we should remember that even such small error keeps us from correct localisation. Therefore, it is not unambiguous that the kNN method – with the smaller horizontal error – should be preferred over the XGBoost algorithm.

5.2 Tests on Access Points from Academic Infrastructure

In the second test, the number of the access points used to build a model was limited to 46 access points from the academic network. The number of trees

Table 2. Results of localisation on 46 access points

Model	HME			ACC
	Median	Mean	80%	
XGBoost(2 models)	2.92	4.13	5.30	0.93
XGBoost(12 models)	2.81	3.99	5.22	0.93
Random forest	-	4.47	-	0.93
kNN	3.13	4.73	5.76	0.91

was reduced to 50 since the reduction in the number of the columns made it unnecessary to use a higher number of the trees.

Table 2 groups the results obtained by the XGBoost algorithm, the kNN method, and the random forests. The same error measures were used as in Table 1. However, not all measures were known for the random forest. Still, we can see that the improvement over the random forest is greater when we use only signals from the academic Wi–Fi network. The XGBoost algorithm also performed better than the kNN algorithm on this smaller dataset. This is especially important because of a reduction of data. In a practical application the reduction can improve the learning time and the obtained results should be more stable than the result from the model that utilises all access points including temporary and mobile ones.

Figure 2 shows the distribution of the horizontal errors on the test data for three algorithms: the XGBoost algorithms with two models for the horizontal location, the XGBoost algorithm with twelve models for the horizontal location and the kNN algorithm. For the kNN algorithm the optimal value of the parameter k was 20. We can see that for the shorter length of input data the XGBoost algorithm provides an improvement over the kNN method. The kNN method fails with the gross errors mostly. The difference for 80th percentile error exceeds half of meter.

The usage of 12 models benefited in the reduction of the horizontal errors. We can observe the reduction of the gross errors specifically. To prove that there is a significant difference between results obtained by the 12 models and 2 models, we performed Wilcoxon's Signed–Rank test for paired scores [9].

For both estimator we created a vector that contains the six **HME** errors presented in Tables 1 and 2.

Wilcoxon's Signed–Rank test rejected the null hypothesis ($p = 0.062500$), which stated that the results obtained by the two estimators were not significantly different, at the 0.1 level. Moreover, the modified test accepted ($p = 0.984375$), at the 0.1 level, the alternate hypothesis that the differences between errors of the XGB(2) and the XGB(12) come from a distribution with median greater than 0. Therefore, the results obtained by XGB(12) model are statistically better.

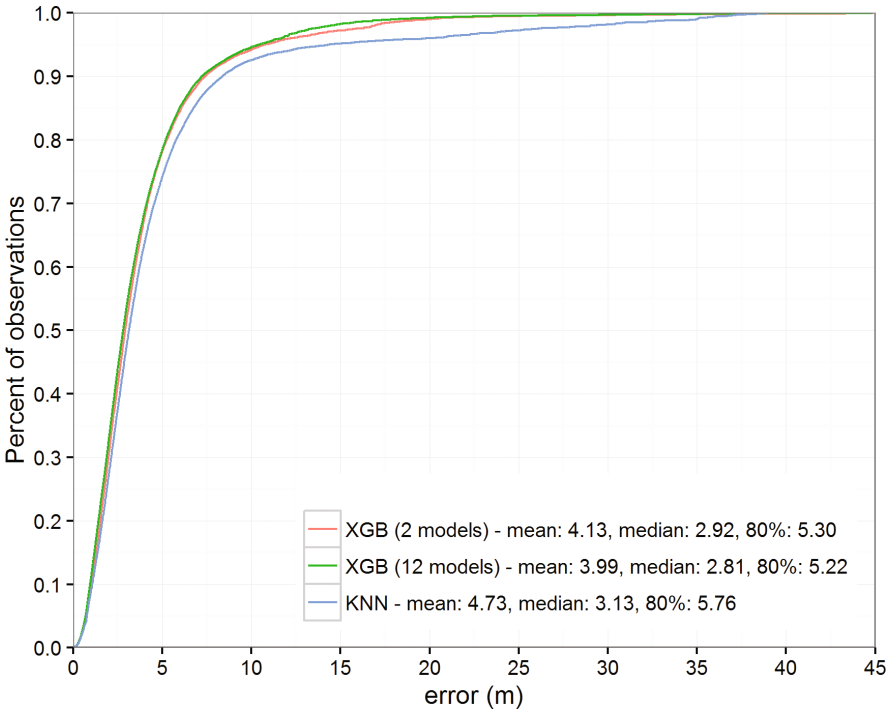

Fig. 2. Results of horizontal localisation on 46 access points

All algorithms – except the kNN method – obtained the same accuracy in the floor detection task. Once again, the classification error was not greater than one floor mostly.

The obtained results are worse than the results obtained for all 570 access points. However, it can be better to use the limited number of the access points in a practical case. The well–known access points are under control and other access points can be switch off without any notification. It may reduce the localisation quality significantly. At the same time a localisation based on the infrastructure can work several years keeping good localisation results [6].

In practical application, XGBoost and Random Forest may be easier to use than kNN method. The implementation itself is not a problem in any case since all algorithms are commonly available in many libraries. However, the prediction in kNN requires constantly referring to the training set, which must be kept available during localisation phase. On the other hand, estimating location with tree-based models is much easier and only requires checking a certain number of logical conditions without the need to access any large volume of data.

6 Conclusions

We have presented the application of the XGBoost algorithm for the localisation task based on the fingerprinting method. In the proposed application schema, we have used one model to estimate a current floor number and 12 models to estimate a horizontal position.

The application was tested on two sets of real data that varied in the length of the input vectors. The proposed algorithm worked similar to the reference method – the kNN algorithm – on longer vectors and better on shorter vectors. For both data sets, the obtained results were better than the published results obtained on the same data sets.

In future work we want to test the algorithm on other data sets and compare it with a wider spectrum of the localisation systems.

References

1. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
2. Chen, T., Guestrin, C.: Xgboost: a scalable tree boosting system. CoRR abs/1603.02754 (2016). http://arxiv.org/abs/1603.02754
3. Dalce, R., Val, T., van den Bossche, A.: Comparison of indoor localization systems based on wireless communications. Wirel. Eng. Technol. **2**(4), 240–256 (2011). http://dx.doi.org/10.4236/wet.2011.24033
4. Dawes, B., Chin, K.W.: A comparison of deterministic and probabilistic methods for indoor localization. J. Syst. Softw. **84**(3), 442–451 (2011). http://www.sciencedirect.com/science/article/pii/S0164121210003109
5. Górak, R., Luckner, M.: Malfunction Immune wi–fi localisation method. In: Núñez, M., Nguyen, N.T., Camacho, D., Trawiński, B. (eds.) ICCCI 2015. LNCS, vol. 9329, pp. 328–337. Springer, Cham (2015). doi:10.1007/978-3-319-24069-5_31
6. Górak, R., Luckner, M.: Long term analysis of the localization model based on wi-fi network. In: Król, D., Madeyski, L., Nguyen, N.T. (eds.) Recent Developments in Intelligent Information and Database Systems. SCI, vol. 642, pp. 87–96. Springer, Cham (2016). doi:10.1007/978-3-319-31277-4_8
7. Górak, R., Luckner, M.: Modified random forest algorithm for wi–fi indoor localization system. In: Nguyen, N.-T., Manolopoulos, Y., Iliadis, L., Trawiński, B. (eds.) ICCCI 2016. LNCS, vol. 9876, pp. 147–157. Springer, Cham (2016). doi:10.1007/978-3-319-45246-3_14
8. Grzenda, M.: On the prediction of floor identification credibility in RSS-based positioning techniques. In: Ali, M., Bosse, T., Hindriks, K.V., Hoogendoorn, M., Jonker, C.M., Treur, J. (eds.) IEA/AIE 2013. LNCS, vol. 7906, pp. 610–619. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38577-3_63
9. Japkowicz, N., Shah, M.: Evaluating Learning Algorithms: A Classification Perspective. Cambridge University Press, New York (2011)
10. Karwowski, J., Okulewicz, M., Legierski, J.: Application of particle swarm optimization algorithm to neural network training process in the localization of the mobile terminal. In: Iliadis, L., Papadopoulos, H., Jayne, C. (eds.) EANN 2013. CCIS, vol. 383, pp. 122–131. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41013-0_13

11. Korbel, P., Wawrzyniak, P., Grabowski, S., Krasinska, D.: Locfusion api - programming interface for accurate multi-source mobile terminal positioning. In: 2013 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 819–823, September 2013
12. Okulewicz, M., Bodzon, D., Kozak, M., Piwowarski, M., Tenderenda, P.: Indoor localization of a moving mobile terminal by an enhanced particle filter method. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2016. LNCS, vol. 9693, pp. 512–522. Springer, Cham (2016). doi:10.1007/978-3-319-39384-1_45