Beyond the RSSI value in BLE-based passive indoor localization: let data speak

Pedro E. López-de-Teruel
University of Murcia
Murcia, Spain
pedroe@um.es

Óscar Cánovas
University of Murcia
Murcia, Spain
ocanovas@um.es

Félix J. García
University of Murcia
Murcia, Spain
fgarcia@um.es

Rubén González
Diwow
Murcia, Spain
ruben@angelplus.es

José A. Carrasco
Diwow
Murcia, Spain
jacarrasco@angelplus.es

ABSTRACT

In this paper we present the results obtained from a large experimental environment that makes use of Bluetooth Low Energy (BLE) as the core technology for a location estimation system. BLE is a common technology for this kind of geopositioning systems, but most of the existing proposals are based on the RSS (Received Signal Strength) value obtained by mobile smart devices from static emitters such as iBeacons or other similar tags. This is not our case, since we adopt a passive approach where monitors obtain advertising frames emitted by mobile BLE beacons with no computing capabilities. In our particular scenario, based on a commercial application of our system, we perform fast but exhaustive training procedures to produce an initial dataset that is then analyzed paying attention to important design parameters. A thorough analysis of the data by means of different data visualization techniques reveals valuable information about the behavior of the emitters, signal characterization, radio coverage and, mainly, possible features that can be employed lately by machine learning methods in order to provide accurate location estimations. This is useful to define a quick and continuous training life-cycle which enables the detection of inconsistent data or failures. Our analysis also suggests that a representation of the observations using alternative features provides similar or even better results than the RSS values in terms of efficiency and support for heterogeneous devices.

CCS CONCEPTS

• **Networks** → **Location based services**; • **Human-centered computing** → **Empirical studies in ubiquitous and mobile computing**;

KEYWORDS

BLE, indoor localization, passive systems, realistic datasets.

ACM Reference Format:
Pedro E. López-de-Teruel, Óscar Cánovas, Félix J. García, Rubén González, and José A. Carrasco. 1997. Beyond the RSSI value in BLE-based passive indoor localization: let data speak. In *Proceedings of MobiQuitous 2018 - 15th EAI Int. Conf. on Mobile and Ubiquitous Systems (MobiQuitous 2018).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

MobiQuitous 2018, Nov. 2018, New York, USA
2016. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION AND RELATED WORK

Over the past twenty years, several radio technologies have been proposed for localization purposes. However, the recent introduction of Bluetooth 4.0 and the Bluetooth Low Energy (BLE) subsystem enables new opportunities. Already supported on a wide range of devices, BLE was designed for ubiquitous computing and the Internet of Things. BLE devices are small, inexpensive and run on batteries for months or years. In some particular scenarios, for practical reasons, it is more appropriate to track valuable assets and people using this kind of small, cheap and off-the-shell devices.

The work presented in this paper was developed making use of an existing localization service which is currently in commercial exploitation. It is placed in a large truck workshop and the main purpose of that application environment is to optimize the repair service and to enhance the service quality by analyzing the required time for each operation and operator. Workers, trucks and valuable equipments are tracked using the localization service based on BLE.

Taking into account our previous background and experience described in [8], we developed a passive indoor localization system based on BLE devices and Raspberry Pi monitors which has several advantages from different points of view: taking into account the economical cost, we are employing 2$ beacons and 30$ monitors; regarding autonomy, beacons have a battery lifetime around 2 years when emitting 1 frame per second; deployment is easy since we make use of a wireless 802.11 network in order to interconnect the monitors with the central server; regularity, since the traffic pattern of BLE advertisements is guaranteed and it can be as frequent as required (several frames per second). In relation to our previous works based on 802.11, as we will show, making use of BLE we obtain much more data in smaller training periods (around 30 minutes for almost 6000 m^2).

As in most fingerprinting based methods, our training phase implies the generation of an initial geo-tagged radio map of signals emitted by the BLE beacons and captured by the monitors. We can find in the literature a good number of research works providing different fingerprinting localization techniques for BLE [2, 4, 5] but, to the best of our knowledge, they are validated usually in small and not realistic scenarios, not taking into account the heterogeneity of the devices involved, and making use of the RSSI value as the unique source of information to infer location. Additionally, they do not use the beacons as the devices to be located, but instead as anchor

devices deployed in the scenario at fixed locations and emitting frames to be processed by the mobile device –usually a smartphone or tablet– to be located. This configuration is not appropriate for certain working environments. Other works make use of devices and configurations similar to ours for other purposes like care of elders and children [6], but they follow an approximation for coarse classification by zones, rather than for position regression. A valuable work which provides insight and experiences from large-scale deployments can be found in [3].

Making use of the trained radio map, composed by thousands of geo-tagged and timestamped RSS signals, the first stage is to analyze which information is really useful. The first contribution of this work are the set of techniques for data visualization that *let data speak*, that is, we display the training data in multiple forms in order to determine which kind of information is more reliable to perform localization. For example, we will show that the RSSI value in our BLE/Raspberry Pi setting is very noisy, but we can still make use of additional information, such as the number of frames received during a given sampling period, or even completely ignore the RSSI absolute value of each capture. The use of these alternative features to perform localization is our second contribution: we will describe up to three additional data representations beyond the classical RSSI value that will prove equally useful to provide precise (in the $2.5 \sim 5$ m range) localization: *binary* vectors built from the mere absence/presence of captured frames; *ternary* vectors built from binary comparisons of RSSI values received from different monitors; and *frame* vectors built from the fraction of total number of received frames with respect to the theoretical maximum.

We finally analyze three different regression techniques on all the defined representations. The first are two classical regression methods without memory, k-nearest neighbour and naive Bayes. The third method adds temporal consistency constraints, being based on hidden Markov models. We obtain state-of-the-art performance in relation to accuracy and estimation error even when the RSSI absolute value is discarded. We plan to make our dataset freely available, as well as the procedures that we have followed, according to the principles of the reproducible science [9].

The paper is structured as follows. Section 2 presents an overview of the scenario and the deployed passive localization system. Section 3 then performs an exhaustive data analysis of our collected dataset, using a variety of useful data visualization techniques. Then, an overview of the different regression techniques that we have employed to perform localization is presented in section 4. Finally, section 5 contains the experimental evaluation of these techniques in several different partitions of the complete dataset, while conclusions and future work are drawn in Section 6.

2 SCENARIO AND LOCALIZATION SYSTEM

Our system is deployed in a truck repair and service shop with an area of 5800 square meters, where 2800 square meters are indoors and the rest outdoors (Fig. 1). Employees, trucks and certain work materials are equipped with BLE devices (*beacons*) which will be tracked by the localization service. They are mainly wristbands, bracelets and tags without any computing capabilities, since they must not interfere with the usual work routines and therefore the use of smartphones or other similar devices is not suitable for this

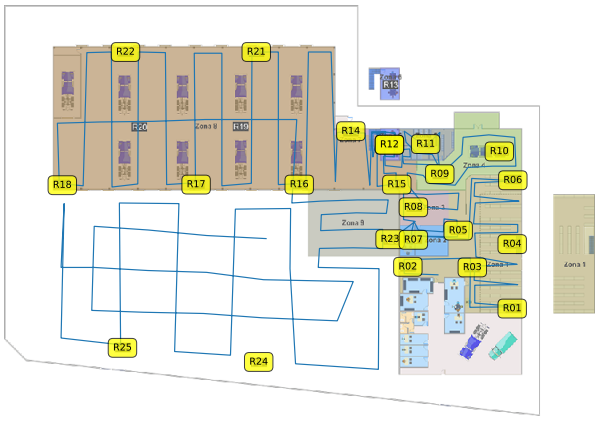

Figure 1: Scenario, training path and monitor positions.

environment. Consequently, there is no specific software component running on the devices other than a BLE emitter transmitting advertisement data on a regular basis. Although the advertising rate can be as frequent as required, beacons typically emit 1 frame per second in order to guarantee a long battery lifetime (around 2 years). Additionally, since every kind of asset to be tracked might require a different type of device, we must assume a wide variety of hardware, antennas, transmitting power and the like.

Most of the logic of the location system is performed by central elements called *monitors*, which must be geographically distributed throughout the environment. We use 25 Raspberry PI 3 units[1], which have many advantages against other alternatives, particularly their small size, low-cost, open source deployment libraries and built-in support for BLE. Every unit continuously runs a monitoring software able to capture BLE traffic and transmit the relevant information via Gigabit Ethernet to a central server. In order to perform its scanning, each monitor scans the different advertisement channels periodically, following a plain round-robin schedule. For each captured packet the only information which is used is the MAC addresses of both the emitting beacon and the monitoring device, the RSSI value, and the corresponding timestamp.

Once that both beacons and monitors have been correctly deployed, a training phase must be performed in order to get the fingerprinting database. Our training process implies the generation of a geo-tagged radio map of signals emitted by the BLE beacons and acquired by the monitors, which will be used as both training data and ground truth to test the regression procedures. Using a mobile app, the operator first defines a set of connected waypoints (training walk shown in blue in Fig. 1) that forms the path to survey the zones of interest. Then the operator takes a set of different BLE devices and distributes them throughout different parts of his body (pockets, backpack, jacket, etc.) and follows the survey path synchronized with the mobile app. Once the survey is finished, the operator can visualize and validate the geo-tagged radio map using the mobile app itself.

[1] Out of which only 23 were fully operative during our tests, located in the positions indicated by yellow rounded boxes in Fig. 1.

Finally, in the operational phase of the system the central server continuosly registers and stores the updated information of captures sent to it by the monitors, while simultaneously runs the localization software responsible for calculating occupancy and positioning information when required. In order to do that, the server provides an API that will be used by higher-level location-based services. In our particular scenario, for example, the system can report where a particular worker or piece of equipment is located just now, the repair time dedicated by workers in each truck, the mean repair time to fix a particular truck fault, or alert with warning messages in real time regarding any service or security condition related to space/time location for any particular beacon, among other similar services.

3 DATA ANALYSIS

3.1 Raw data

One of the advantages of the passive approach -i.e., using the BLE beacons as mobile elements to be localized, instead of as fixed anchors- is that we can obtain lots of training data in short periods of time. With the operator carrying many of these devices simultaneously during the training, as described in the previous section, and by controlling the frame emission period of the devices, the number of monitored samples gets easily incremented. In this section we describe the original dataset of raw samples in which we based all our data analysis and posterior experimental results.

We used up to 11 BLE devices of three different types: 3 BlueBar beacons (*BBnn*), 7 Nordic beacons (*GT_nnn*), and 1 iPod (*IPOD*) executing a dedicated BLE app, whose advertising periods were adjusted to 0.75, 1.0 and 0.2 seconds, respectively. The operator carried these beacons in different parts of his body (front, back, upper and lower pockets, different places inside and outside a backpack, etc.), while walking through the guided training path shown in blue in Fig. 1. The whole controlled training time interval took just 2116 seconds (approximately 35 mins), during which a total of 84072 frames emitted by the 11 devices were captured by the 23 installed and fully operative monitors (*Rnn* in Fig. 1). Table 1 shows the total number of samples obtained per each device/monitor (dev, mon) pair. The corresponding distributions of values for the obtained RSSI and elapsed time between successive samples captured for a same (dev, mon) pair is shown using box plots[2] in Fig. 2 below the table, for one device of each of the three types.

From the RSSIs plot we can observe a great variability in the strength of the signal depending not only on the device but, all about, in the monitors themselves, with some monitors (i.e. R09 or R15) clearly receiving weaker signals, mainly due to physical obstacles near their respective locations in the scenario. The much larger observed variability in elapsed time between successive samples is directly related to the different frame emission periods of each type of device, and also to the fact that, of course, not every place in the map is reached by every monitor. Of course, this variability is also closely correlated with the position of the beacon, and it is this correlation that is typically exploited by machine learning based methods as the ones that we use in our localization system. This correlation between the strength of the signal and the distance to the monitor is also illustrated in Figs. 3 and 4, which also show the typical noise of the BLE RSSI signal (both in time and space) which makes the localization problem harder. Again, some monitors have less geographical scope than others due to physical occlusions (compare, for example, R01 *vs.* R10 in Fig. 4).

3.2 Processed data

As practically all machine learning techiques work with vectors as input, the next step we had to do was to perform a uniform sampling of the whole set of 84072 (*dev*, *mon*, *timestamp*, *rssi*) obtained tuples in order to vectorize them over time. The exact sampling interval used clearly determines here the final number of vectors obtained, as well as the quality of the obtained information. In this sense, two important parameters when generating both the training and test vectorized datasets from the continuous in time input data are the following:

- Persistence T_p, which is the time interval during which the influence of each RSSI sample is "lengthened": values are filled forward in time with the last available value up to this maximum tolerance, or until a new, fresh sample is received for the same (dev, mon) pair. For monitors that did not receive a sample for the given device in the last T_p seconds, the corresponding vector elements are filled with a minimum value of -120 dBm[3].
- Uniform sampling interval T_s: the larger, the smaller number of vectors, but also the greater number of input samples to consider when computing a component in the vector, which will be hopefully more reliable.

Through the rest of this paper, except where otherwise specified, we have considered a value of $T_s = 1$ seconds, with a persistence value of $T_p = 5$ seconds. These values are justified because all the devices emmit a new frame in, as much, 1 second, though some of these frames can eventually get lost in the capturing process, which makes a value of $T_p > 1$ necessary. Following this sampling policy, and being $M = 23$ the number of monitors, we generated up to four different types of sampled vectors:

- Received signal strength intensity (*rssi*) M-dimensional vectors, where each individual component corresponds to a RSSI measure in *dBm*.
- Simple M-dimensional binary vectors (*binary*), which only preserve information about presence/ausence of captured frames during the sampling interval, discarding any original RSSI, time or number of frames information.
- Frame proportion M-dimensional vectors (*frames*), which store in each component the fraction of total number of received frames with respect to the maximum emitted by the beacon for a given sampling period. The resulting value will be in the interval $[0.0, 1.0]$.
- Finally, larger M'-dimensional *ternary* vectors built from RSSI magnitude comparisons between all the $M' = \binom{M}{2} = \frac{M*(M-1)}{2}$ combinations of the M monitors by pairs. In our case $M' = \binom{23}{2} = 253$. The name *ternary* comes from the fact that each individual component in these vectors will always

[2]The limits of the box plots (out of which the data are considered outliers) are placed in the 1% and 99% percentile, respectively.

[3]This corresponds to an arbitrary lower threshold smaller than the overall minimum observed RSSI value, -113 dBm in our study.

Monitor Device	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10	R11	R12	R13	R14	R15	R16	R17	R18	R21	R22	R23	R24	R25
BB6A	105	138	105	50	616	47	23	189	3	421	208	74	20	285	8	279	245	88	166	246	92	50	47
BB87	279	581	178	147	1071	124	89	343	73	970	584	343	70	559	111	659	547	306	442	524	328	261	334
BBFC	308	654	224	201	1139	145	165	369	94	1000	645	380	93	586	142	684	580	366	480	558	384	296	362
GT_034	125	266	81	94	563	56	45	169	14	462	290	133	35	290	38	342	268	127	180	265	156	95	85
GT_055	27	40	32	15	191	20	4	57	0	139	49	28	14	84	2	114	92	34	63	85	25	16	13
GT_056	83	182	56	57	392	25	28	128	11	326	193	80	21	187	10	235	170	106	147	184	81	51	57
GT_057	91	223	93	81	548	49	51	151	15	382	250	113	25	239	20	307	221	104	177	223	133	43	55
GT_1002	142	351	125	119	747	75	68	202	38	612	392	202	41	376	60	435	345	187	270	313	204	93	86
GT_1003	158	343	95	112	678	68	65	208	33	586	347	209	42	349	52	412	347	211	278	339	201	123	167
GT_1005	123	267	95	89	609	57	52	187	30	494	285	147	35	290	51	324	292	148	241	270	172	64	95
IPOD	1128	2252	623	696	4323	468	374	1322	221	3614	2351	1298	334	2424	369	2600	2178	1272	1761	2021	1335	805	970

Table 1: Total number of captured frames per (dev, mon) pair.

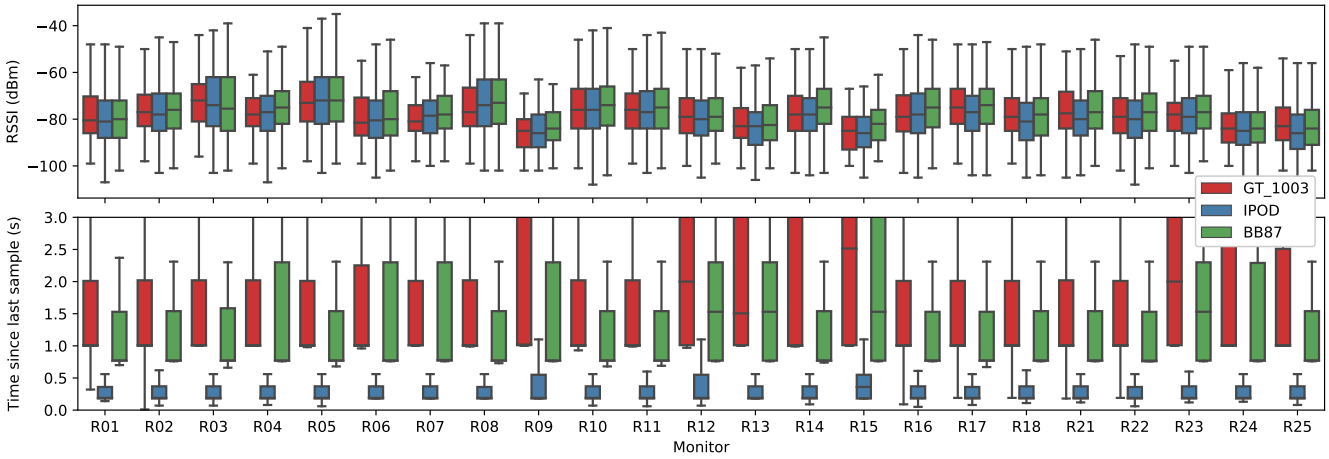

Figure 2: Distribution of RSSI (top) and last sample time (bottom) values for three devices of different types.

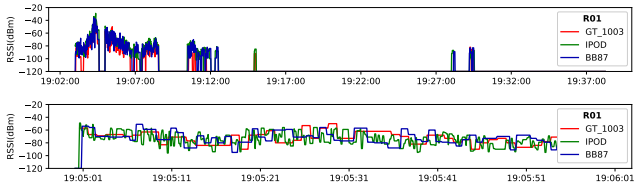

Figure 3: RSSI *vs.* time for a given monitor (R01) and three different type of devices. Top: Whole training interval (35 minutes). Bottom: Zoomed one minute interval.

take one of the three values $\{-1, 0, +1\}$, depending on the magnitude comparison between the RSSI values corresponding to the compared values [7]. The values -1/+1 stand for dominance of the first or second compared monitor, while 0 means equality -typically, because neither of both monitors captured any frame for the given device and pair of considered monitors during the corresponding sampling interval-. The effect of one of these monitors comparison is illustrated in Fig. 5.

Fig. 6 (top and center) shows graphically the sequence in time of obtained *rssi* and *frames* vectors for three devices of the three different classes[4]. The total number of *rssi* vectors is 2116, as corresponds to a value of $T_s = 1$, while we only got 2116/5=424 *frames* vectors, because we had to use a larger sampling period of $T_s = 5$ seconds to get more reliable values of proportion of captured frames per interval for the slower emitting Nordic devices. On the bottom part of the figure we also show the evolution in time of the number $n_a(v_t)$ of *active* components for each vector v_t (i.e. values $\neq -120$ dBm for *rssi*, and $\neq 0$ for *binary* and *frames* vectors, as corresponds to no frames captured for the given (*dev*,*mon*) pair in the corresponding sampling period). Of course, the persistence value T_p directly influences this n_a number in all vector types: the longer T_p, the more information will be captured by more active components in the vectors, but also the less precise will be the ground truth location information, due to the operator movement. The relative increments of n_a as we increase T_p are shown in Fig. 7, and its decisive influence on the quality of the position regression will be demonstrated in Section 5.

[4] *Binary* vectors are not shown, as they are simply binarized versions of the corresponding *rssi* vectors, while *ternary* vectors are not shown either, due to lack of space and also because they do not have such a clear visual interpretation.

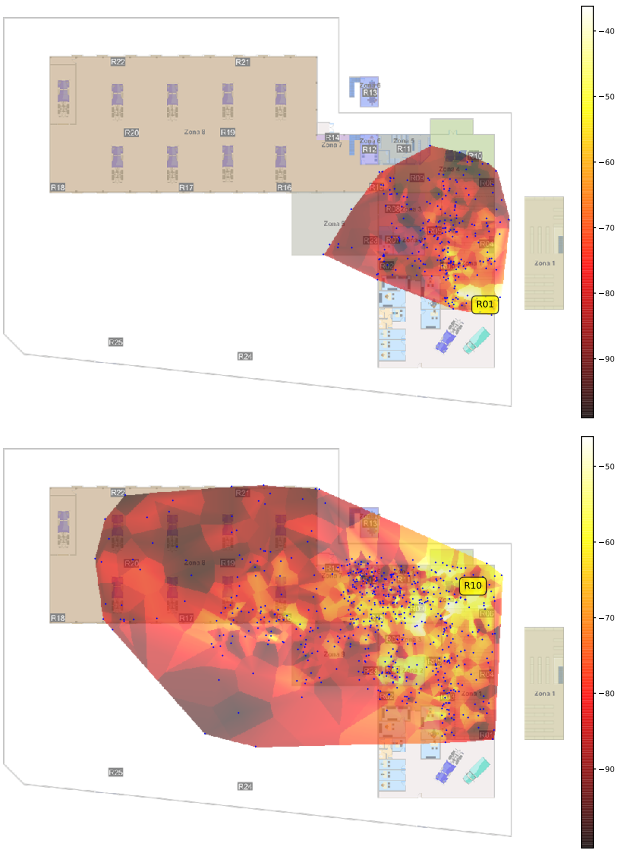

Figure 4: Interpolated heatmap of RSSI superimposed on the scenario, with corresponding sampling points: (*BB87,R01*), 279 samples (top). (*GT_1003,R10*), 586 samples (bottom). Slight random noise (up to 2 meters) has been added to sampling positions to improve readability.

Finally, we also considered important to study the robustness of the RSSI based representation to device heterogeneity. Fig. 8 serves us to visually represent this issue. The shown graphs represent the evolution over time of the *monitor dominance* for the three classes of devices. In this representation, the used quantity of color for each column is proportional to the observed RSSI signal by each monitor for the corresponding device at that precise instant of time. Though there are certainly differences of detail between different devices, there is a clear tendency to be observed with similar relative strengths by the same monitors for the three types of devices. It is these small differences that the position regression methods will have to absorb to deal with the heterogeneity of devices.

3.3 Discussion

From the data analysis performed in this section we can conclude that, in general, RSSI is very noisy in both the time and space domains, being this noise even more pronounced in a BLE setting than in the more classical 802.11 one. But we have also visualized how, by adjusting the values of important design parameters –such as the persistence T_p and the sampling interval T_s–, as well as adequately

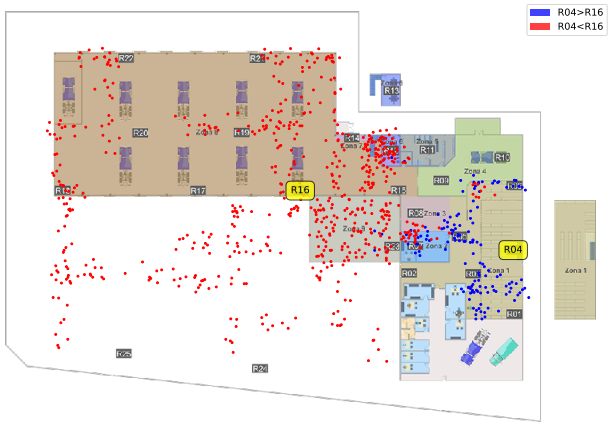

Figure 5: Individual component of ternary vectors for device *GT_1005*, corresponding to the pair of monitors (*R04,R16*). Red means -1, blue +1.

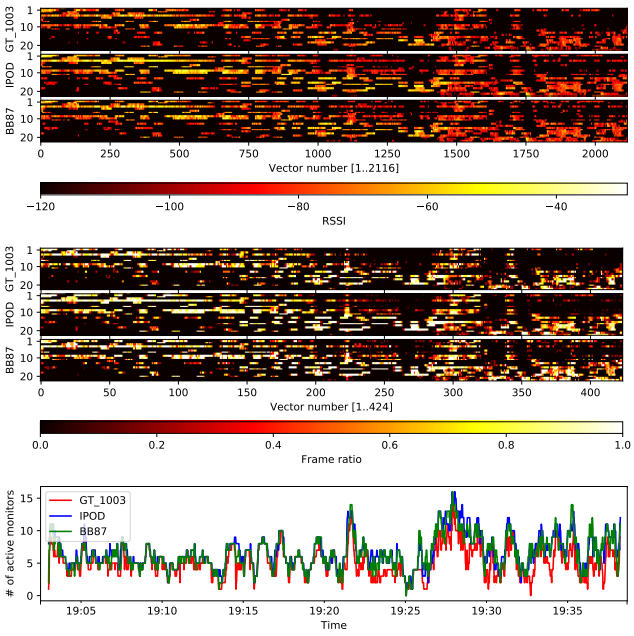

Figure 6: Set of *rssi* (top) and *frames* (center) vectors for the training time interval. Corresponding number of *active* components *vs.* time (bottom).

preprocessing data with techniques specifically designed to deal with device heterogeneity –such as just comparing signal strength between monitors, or measuring the proportion of missed frames for a given sampling interval, in both cases discarding RSSI–, we can still get different vectorial data representations that, as will be shown in the next sections, will prove perfectly valid to obtain good localization results, specially when boosted by using time consistency in the regression procedures.

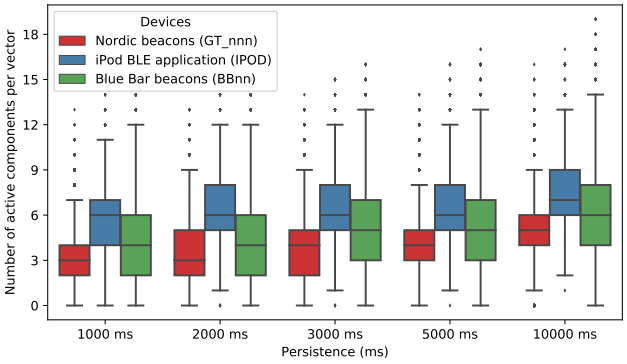

Figure 7: Distribution of number of active monitors per vector as a function of T_p persistence (in ms), for the three different kind of devices.

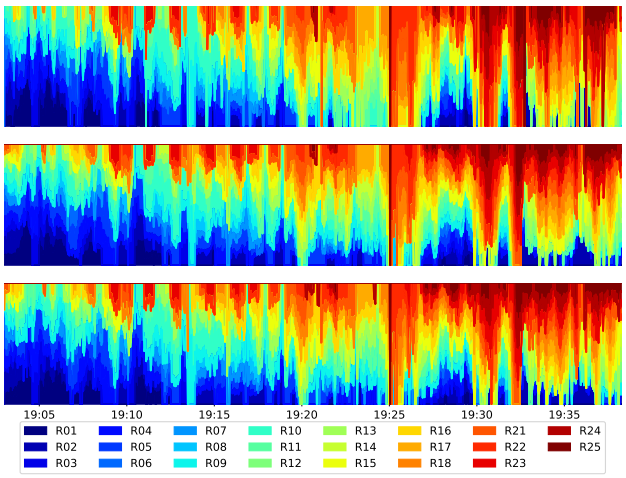

Figure 8: Monitor dominance for devices *GT_1003* (top), *IPOD* (center) and *BB87*(bottom).

4 REGRESSION METHODS

In order to verify the validity of using the different vector types defined in the previous section to solve the localization problem, we will make use of three different kind of regression methods. The first two are two classical machine learning methods, k-nearest neighbour (*kNN*) and naive Bayes (*NB*), that do not take into account temporal consistency, and that therefore work by considering both the input training and test vectors as independent and identically distributed. The third method adds temporal consistency constraints, which basically exploit the fact that a given device can't move long distances in short periods of time. Our choice for this is to use a classical hidden Markov model (*HMM*) over a discrete grid space of configurable cell size defined over the scenario map. In what follows we will describe the details of our tested implementations.

4.1 K-nearest neighbour regression

The use of the *kNN* model for regression is justified by its good performance when the size of the training data is limited. Though obviously very simple, it is a very powerful alternative that tends to give very good results in situations of homogeneity of devices, and that does not require a previous grid discretization, as it can directly work with ground truth positions (x, y) over the map. The most important free parameter of this model is the number of neighbours k, which can be cross validated as we will describe later in the results section (Fig. 12). As of the metric to use, we will use the usual simple euclidean distance (though certainly many more alternatives could be used; see, for example [8] for other meaningful possibilities).

4.2 Naive Bayes on grid of positions

One problem with *kNN* is that it is not trivial to get a conditioned probability density $p((x, y)|v_t)$, especially when the number of training samples is relatively low with respect to the input space dimensionality. In our case, the input measurements vector dimension is directly related to the number of monitors M, while the available number of vectors for a given location is necessary low, due to the fast training period and large dimensions of the scenario (remember, slighty more than half an hour walk through a 5800 m^2 scenario). Fig. 9 shows, for example, the number of available training vectors for a particular subset of three devices in each of the grid positions resulting of dividing the whole scenario in square cells of 3×3 meters. For completeness, we also show in that map circular marks with different colors for different devices in their corresponding (x,y) sampling positions (adding a small random noise on it to avoid overlapping and improve visibility), and with radius proportional to number of active monitors for the corresponding vector. More importantly, the figure also shows the total number of vectors sampled in each grid cell. As can be seen, the total number of samples per cell is relatively low in many cells (though obviously it would be multiplied by a factor of 11/3 if using all the available beacons).

In order to get a reliable a posteriori probability density $p((x, y)|v_t)$ (with v_t being any measure vector of the types described in Section 3 above) from such a low number of samples per cell, we will use the simple –though still powerful and very popular technique among the machine learning community– naive Bayes model (*NB*). *NB* models make an assumption of independence among the components (*features*) of the input vectors. In our case, this assumption is clearly acceptable only because, once a position (x, y) on the map is given and kept fixed, the random variables corresponding to vector components associated to different monitors can be certainly assumed as independent. This is known as conditional independence [1], and will allow us to compute a reasonable approximation of the a posteriori probability for each given cell position in the grid using simple histograms. By choosing adequate bins size and extents (depending on the chosen vector types), we will show in the results section that this method is suitable for all the classes of features mentioned in Section 3.

NB models will therefore be completely defined by a list of probability tables, one for each (x, y) point on the grid. Each table will

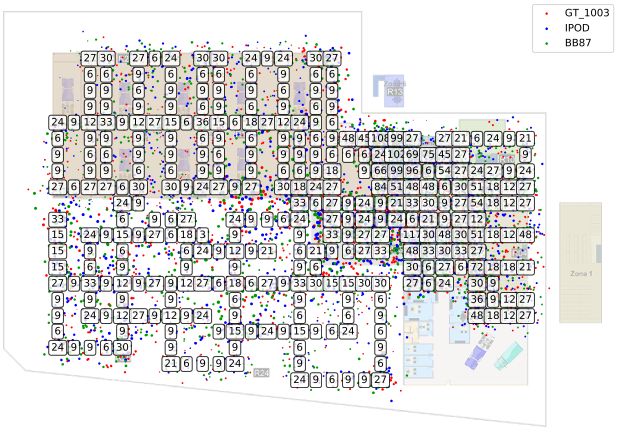

Figure 9: Number of samples per grid cell (vectors positions also shown, size proportional to number of active monitors; 6348 total vectors).

be characterized by a $N \times B$ dimensional matrix, being N the dimension of the input vectors (i.e., the number of monitors M for *rssi*, *binary* and *frames* vector types, or $M' = \frac{M*(M-1)}{2}$ for *ternary* vectors). As for the number of bins B, it will be two ($\{0, 1\}$) and three ($\{-1, 0, +1\}$) for *binary* and *ternary* vectors, respectively, and definable at will in the case of *rssi* and *frames* vectors, by dividing the range of valid values ($[-120; -20]$ dBm and $[0.0; 1.0]$, respectively) in the desired number and extents of intervals. Given these definitions, the computation of each element of these matrices is performed by usual relative frequency counting over the whole set of traning vectors, independently for each vector component as corresponds to the conditional independence assumption.

Another important parameter in the probabilistic model is the grid resolution. Of course, there is a balance here between maximum achievable position accuracy and obtaining a reasonable number of samples per grid point to elaborate the probabilistic models. Also related to it, and given that the number of samples per grid point can be relatively low in a fast training on a large scenario like ours, we also define a maximum distance threshold that will be used to consider a sample when computing the model at a point (x, y) of the given grid. By default, 5.0 meters will be used, which will result in taking into account not only the samples falling in the same grid point, but also some of the ones falling on any of the adjacent cells. Of course, the higher this threshold, the greater number of samples are taken into account in updating the corresponding histograms, but at the cost of a slightly reduced location accuracy. Finally, we also introduce a minimum probability for each element of the histograms to avoid situations of zero probability which could lead to possible numerical issues in certain cases.

Once the a posteriori probability $p((x, y)|v_t)$ has been computed for each cell using the histograms model, final regression can be performed by choosing between a MAP decision (grid point with *maximum a posteriori* estimated probability) or mathematical expectation (weighted average of positions using the estimated vector of probabilities). We opt for this second possibility, which tends to soften the obtained regression results. It should be emphasized, though, that it is the ability of estimating the $p(\cdot)$ density function, rather than this possibility of direct regression using the *NB* method, which will prove absolutely essential to be able to apply the *HMM* technique described in the following section.

4.3 Hidden Markov model

This method extends the previous static probabilistic model to introduce dynamic estimation *with history*, which takes time constraints into account. We implement a classical *HMM* that relies on a probabilistic *movement model* where, given a $(x(t), y(t))$ position at instant t, assigns a probability to each grid position $(x(t+1), y(t+1))$ which is inversely proportional to the traversed distance [10]. In our case, the state transition matrix for this movement model is computed using an exponential fall of the probability to an adjacent state, parametrizable by a σ value in meters. Again, a minimum probability value is assigned to every grid position to avoid numerical errors caused by zero probabilities. In the results section we will illustrate that values of σ between 3.0 and 5.0 meters make perfect sense for $T_s = 1$ second, and in fact the influence of σ on the overall regression results will not prove that important.

The *HMM* model then basically combines the current position estimation with a *measurement model* just as the one described in the previous subsection, assigning a posterior distribution $p((x(t+1), y(t+1))|v_t)$ to every (x, y) position in the grid. In fact, this HMM method exhibits a similar behavior to the Kalman filter and its different extensions (such as EKF, UKF of particle filters [1, 10]), but working with a discrete state space (the grid) instead of using continuous variables.

5 EXPERIMENTAL RESULTS

5.1 Datasets

In this section we describe several experiments that we performed to evaluate the regression results obtained when testing the three methods described in section 4 on the different vector representations described in section 3. In order to do it, we first define four different training/test partitions of our data (see Table 2):

- Dataset A: Typical calibrated (homogeneous devices) case, using only beacons of one type (Nordic *GT_nnn* devices), all functioning correctly, for both training and test subsets.
- Dataset B: Same as A, but testing with a poor functioning beacon (Nordic *GT_055* device; this device was deliberately put in a pocket together with a metallic object -a key ring- which attenuated the emitted BLE frames).
- Dataset C: Typical uncalibrated (heterogeneous devices) case, in which the training set was obtained using the IPOD device, but test was performed by using Nordic *GT_nnn* beacons.
- Dataset D: A different heterogeneous case, in which the training and test were of the Blue Bar (*BBnn*) and Nordic (*GTnn*) devices classes, respectively.

As discussed in subsection 3.2, an important issue for a given training or test vector is its number of active components $n_a(v_t)$. Given that all the regression methods base their operation in using mainly the active components (think of this number as directly related to the quantity of information provided by a given measurement vector), Fig. 10 also shows the total amount of training vectors

	Dataset A	Dataset B	Dataset C	Dataset D
training devices	GT_034, GT_056, GT_057, GT_1002, GT_1003	GT_034, GT_056, GT_057, GT_1002, GT_1003, GT_1005	IPOD	BB6A, BB87 BBFC
testing devices	GT_1005	GT_055 (bad functioning)	GT_034, GT_056, GT_057, GT_1002, GT_1003, GT_1005	GT_034, GT_056, GT_057, GT_1002, GT_1003, GT_1005

Table 2: Different dataset partitions (A, B, C, D) used for the experiments (see text).

retained after removing those which do not have a minimum number of active components, n_a^{min}. We can clearly appreciate that, of course, as we increment this value, we end up discarding a greater percentage of training (and subsequently testing) vectors. But, as we will see in what follows, the higher this value, the lower will also consequently be the obtained regression error of all methods. The price to pay will be, of course, that when discarding training vectors we are effectively reducing our training data, while when a given test vector is discarded it means that we must give up trying to perform the regression, at least until a new measurement vector with the minimum number n_a^{min} of active elements is obtained.

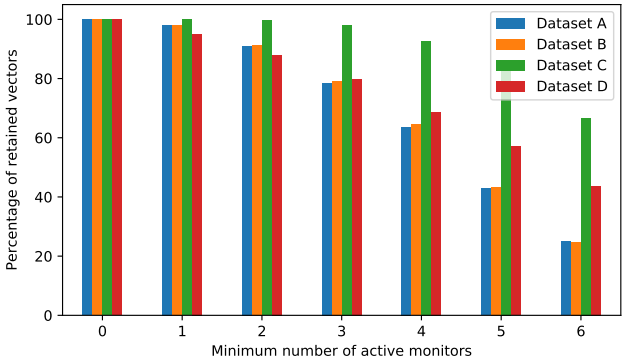

Figure 10: Percentage of preserved vectors vs. minimum number of active monitors required per vector (n_a^{min}), for all four datasets (training devices).

5.2 Study of parameters

We will start studying the influence of n_a^{min} in the regression results. In order to do it we will employ the simplest *kNN* method, using $k = 3$ neighbours and the euclidean distance on *rssi* vectors on dataset A (see Fig. 11). We can see that, as we advanced in the previous subsection, the greater the demanded n_a^{min}, the better the results, though this progressive improvement somehow diminishes for values of $n_a^{min} > 4$. Given also that for $n_a^{min} > 4$ the proportion of discarded vectors also dramatically increases (Fig. 10), in the rest of the experiments we will fix this number of required minimum active components by vector to four. Though not shown in this paper due to lack of space, this value also leads to similar results when testing the rest of vector types and regression methods.

We study now the influence of k in the *kNN* method. Again we use *rssi* vectors on dataset A and show the obtained regression error

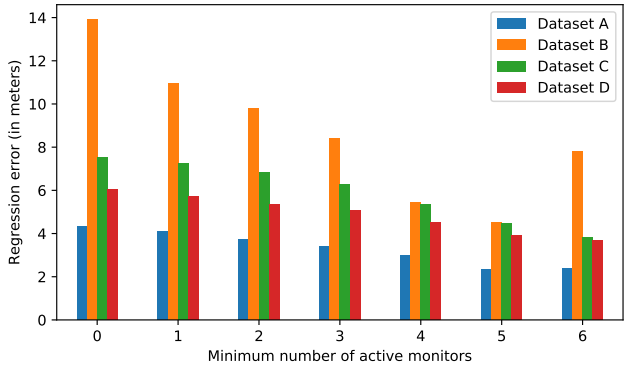

Figure 11: Regression error for increasing number of minimum active monitors (n_a^{min}) allowed in vectors.

in meters for different values of k in Fig. 12. We see that best results are obtained for $k = 5$ though, in fact, any value between 3 and 9 could also be equally acceptable.

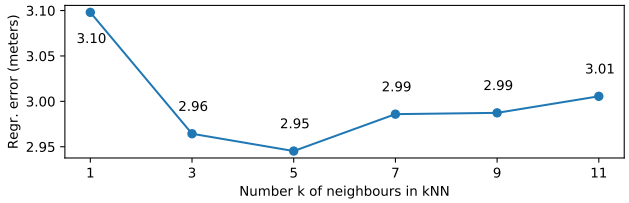

Figure 12: Positioning error for different values of k.

Another adjustable parameter that deserves attention is the σ parameter in the *HMM* model. Remember that this parameter is related to the uncertainty in the movement model described in subsection 4.3: greater values allow for faster movements of the beacons, thus assigning more relative weight to the current measure than to former state (i.e., position). Fig. 13 graphically shows this influence in the final obtained positioning error. For this particular graph we used dataset A, a minimum number of monitors $n_a^{min} = 4$ and *binary* vectors, though, again, the results are qualitatively similar when we test the rest of vector types described in subsection 3.2.

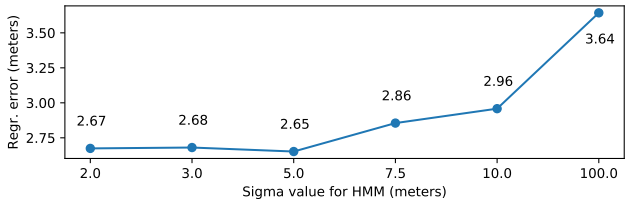

Figure 13: Positioning error for increasing number of σ in HMM.

One interesting thing to note in this graph is that in the extreme case of $\sigma = 100$ we are allowing any possible instant movement to

anywhere in the scenario –i.e., we are effectively discarding temporal consistency, assigning all credibility to the current measure in each instant $t-$. A low value of σ of, say, 2 or 3 meters, on the contrary, only allows effective movements of a beacon from one grid cell to an adjacent one in the following instant (i.e., in $t_{i+1} = t_i + T_s$; remember from subsection 3.2 that in our case T_s had a value of one second when using *rssi*, *binary* and *ternary* vectors, and five seconds for *frames* vectors). This way, by increasing σ from these lower values to extremely large ones we can observe the gradual influence of adding temporal consistency to the regression models. It is remarkable that the obtained regression error can be lowered by approximately 1 meter (from 3.64 to 2.65, using $\sigma = 5.0$) by taking this kind of temporal consistency into account, in contrast to the more naive possibility of regressing the position for each new measure from scratch, as many classical fingerprinting approaches do.

5.3 Study of vector representations

We will now compare the effect of using vector representations alternative to the classical *rssi* one. Fig. 14 shows the distribution of positioning errors for several alternative representations, again on dataset A, and using the simple kNN model. For each histogram, a yellow line marks the mean positioning error, while a green one marks the median of the distribution. It is remarkable that though *rssi* gets the best results ($\sim 3\ m$), the alternative representations which effectively discard the RSSI values, taking into account only the presence/comparison/counting of frames in the respective sampling intervals, achieve very similar results. This is specially the case in *binary* and *ternary* vectors, with *frames* vectors achieving worst, but still acceptable, positioning errors in the $\sim 5\ m$ average range. This is quite interesting, given that it is precisely the variability of the RSSI between different types of devices what makes the positioning problem for heterogeneus devices hard [11]. Note also that though these mean and median positioning errors are kept in average relatively low, still very large positioning errors (say $> 10\ m$) still appear in the histograms. It is this kind of errors the ones that tend to be absorbed by temporal consistency in the *HMM* model.

As for the geographical distribution of these positioning errors, it is shown in Fig. 15 for the case of *rssi* vectors (not shown for the rest of vector types due to lack of space, though it is again qualitatively similar). Observe that positioning error is clearly smaller in the interior of the building, where more training data were available (the training walk was lighter and more spaced outdoors, see Fig. 1).

5.4 Global results

Table 3 summarizes the mean positioning error for the three regression procedures described in section 4 tested on all vector representations over the four proposed datasets partitions. For each dataset, the best results are highlighted in bold. Clearly, the most difficult task is to correctly localize the malfunctioning beacon (*GT_055*), for which most of the emitted frames were lost (see corresponding row in Table 1) due to occlusion of the metallic key chain that was in the same backpack pocket when performing the data gathering walk. Best results are also logically obtained for dataset A, (training

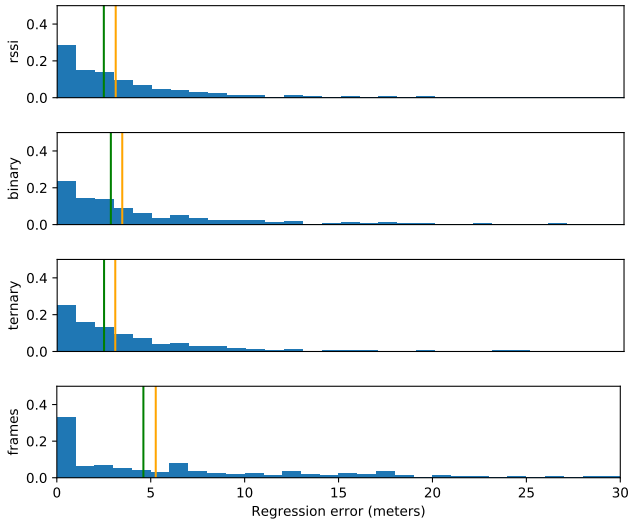

Figure 14: Regression error distributions for the four studied vector types.

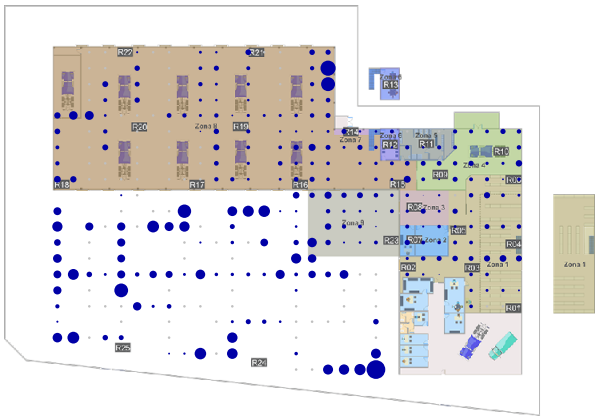

Figure 15: Geographic distribution of errors for *rssi* vectors .

and test using same type of devices, i.e. homogeneous case). In any case, if we look at the global results, there are a few considerations that are worth remarking:

- Many combinations of vector representations and regression methods lead to location errors in the range $3 \sim 5$ meters, with some of them even below, for the easiest case of homogeneous devices.
- In general, temporal consistency introduced by *HMM* always tends to improve the regression results by $1 \sim 1.5$ meters.
- Though the *rssi* representation behaves the best, interestingly, some of the other three representations that discard this measure can achieve very similar results. See, for example, the behaviour of *binary*, *ternary* and *frames* representation on the homogeneous case (dataset A, *HMM*), or the simple *binary* representation for the harder datasets B and

	Dataset A			Dataset B			Dataset C			Dataset D		
	kNN	NB	HMM	kNN	NB	HMM	kNN	NB	HMM	kNN	NB	HMM
rssi	2.96	3.43	**2.42**	5.44	5.94	7.18	5.37	7.01	5.43	4.52	5.13	**3.90**
binary	3.30	3.65	2.65	5.53	**5.13**	5.74	6.34	6.07	**4.87**	5.36	5.38	4.27
ternary	3.10	3.65	3.00	5.69	5.86	5.74	4.88	6.21	4.90	4.66	5.25	4.31
frames	4.85	4.65	2.63	6.80	7.61	5.38	7.37	7.54	5.86	7.02	6.46	4.71

Table 3: Results for the three regression methods on all tested combinations of datasets / vector types.

C. *Ternary* vectors also behave close to optimal in datasets C and D (*HMM*), which represent the heterogeneous device cases (remember that ternary vectors were specifically designed to deal with the problem of locating heterogeneous types of devices [8, 11]).

- The *frames* representation consistently gets the worst results in all datasets. Remember, though, that to obtain reliable component vector values when counting the number of frames of a given period, we had to augment the sample interval to $T_s = 5$ seconds, due to the fact that most of the devices (all the Nordic *GT_nnn* ones) only emitted one frame per second. Given that the operator covered the whole 5800 m^2 scenario in just over half an hour at a normal walking speed, clearly he had to move about $4 \sim 5$ meters during many of these sample intervals. This, of course, clearly affected the quality of the ground truth positions attached to the training examples, and divided the total number of vectors by a factor of five (compare Figs. 6 top and center). We can conclude to this respect that, in order to successfully use this alternative representation, we should perform slower trainings or, alternatively, faster frame emission periods for the devices[5] (just like the *IPOD* device does; see Table 1).

6 CONCLUSIONS

In this paper we have performed an exhaustive experimental analysis in a large, commercial and realistic scenario of BLE based indoor localization. We have emphasized the main advantages of our adopted approach, based on mobile beacons and passive monitorization, in contrast to the opposite, more classical approach of performing the localization on mobile smart devices with available computing capabilities. These advantages are mainly the much lower implementation cost, as well as the much lighter training efforts needed to get a reliable training dataset for fingerprinting.

We have also demonstrated the importance of a previous good data analysis, based on adequate visualization techniques which, in our case, even led us to successfully try alternative data representations to the classical RSSI values, getting comparable positioning results, specially when using temporal consistency through HMM models. These alternative representations can help in correctly addressing the problematic issue of device heterogeneity that characterizes many localization systems both passive and active. Furthermore, we have also shown how a thorough analysis of the sampled data can provide fast, helpful and intuitive visual feedback about the nature and issues of the training process, as well as the influence that different implementation details can have in the quality of the positioning results. We strongly believe that this kind of analysis can be an exceptional tool to aid in the deployment cycle of commercial indoor positioning systems based on radio fingerprinting.

As a statement of direction we are currently working on elaborating a much larger dataset, including different days of operation, as well as longer duration training walks, to make it available to the research community as a public dataset and further test on it additional machine learning techniques. Introducing the Viterbi algorithm [1] to further improve the beneficial effects of temporal consistency, as well as extending our study to the problem of location by zone classification (instead of position regression) will also be our focus of interest in the near future.

[5]Though this last option would of course negatively affect the battery life of the beacons.

ACKNOWLEDGMENTS

This work was supported by the Spanish MINECO, as well as by European Commission FEDER funds, under grant TIN2015-66972-C5-3-R.

REFERENCES

[1] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag, Berlin, Heidelberg.
[2] David Contreras, Mario Castro, and David Sánchez Torre. 2017. Performance evaluation of bluetooth low energy in indoor positioning systems. *Transactions on Emerging Telecommunications Technologies* 28, 1 (2017).
[3] Xenofon Fafoutis, Atis Elsts, Robert Piechocki, and Ian Craddock. 2018. Experiences and Lessons Learned from Making IoT Sensing Platforms for Large-Scale Deployments. *IEEE Access* 6 (2018), 3140–3148.
[4] Ramsey Faragher and Robert Harle. 2014. An analysis of the accuracy of bluetooth low energy for indoor positioning applications. In *Proceedings of the 27th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+'14).* 201–210.
[5] Ramsey Faragher and Robert Harle. 2015. Location fingerprinting with bluetooth low energy beacons. *IEEE journal on Selected Areas in Communications* 33, 11 (2015), 2418–2428.
[6] Chuan-Bi Lin, Yung-Fa Huang, Long-Xin Chen, Yu-Chiang Chang, Z-Ming Hong, and Jong-Shin Chen. 2018. A Low-Costed Positioning System Based on Wearable Devices for Elders and Children in a Local Area. In *Asian Conference on Intelligent Information and Database Systems.* Springer, 324–332.
[7] Pedro E. Lopez-de Teruel, Oscar Canovas, and Felix J. Garcia. 2017. Using Dimensionality Reduction Techniques for Refining Passive Indoor Positioning Systems Based on Radio Fingerprinting. *Sensors* 17, 4 (2017).
[8] Pedro E. Lopez-de Teruel, Felix J. Garcia, and Oscar Canovas. 2018. Practical passive localization system based on wireless signals for fast deployment of occupancy services. *Future Generation Computer Systems (In press)* (2018).
[9] M. Munafo, B. Nosek, D. Bishop, K. Button, C. Chambers, N. Percie du Sert, U. Simonsohn, E. Wagenmakers, J. Ware, and J. Ioannidis. 2017. A manifesto for reproducible science. *Nature Human Behaviour* 1, 1 (2017).
[10] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. 2005. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents).* The MIT Press.
[11] S. Yang, P. Dessai, M. Verma, and M. Gerla. 2013. Freeloc: Calibration-free crowdsourced indoor localization. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM).*