Human SLAM, Indoor Localisation of Devices and Users

Wouter Bulten
Artificial Intelligence Department
Radboud University
Nijmegen, The Netherlands
Email: wouterbulten@mai-ai.org

Anne C. van Rossum
Almende B.V. & DoBots B.V.
Rotterdam, The Netherlands
Email: anne@dobots.nl

Willem F.G. (Pim) Haselager
Donders Institute for Brain, Cognition and Behaviour
Radboud University
Nijmegen, The Netherlands
Email: w.haselager@donders.ru.nl

***Abstract*—The indoor localisation problem is more complex than just finding whereabouts of users. Finding positions of users *relative* to the devices of a smart space is even more important. Unfortunately, configuring such systems manually is a tedious process, requires expert knowledge, and is sensitive to changes in the environment. Moreover, many existing solutions do not take user privacy into account.**

We propose a new system, called *Simultaneous Localisation and Configuration* (*SLAC*), to address the problem of locating devices and users relative to those devices, and combine this problem into a single estimation problem. The SLAC algorithm, based on FastSLAM, is able to locate devices using the received signal strength indicator (RSSI) of devices and motion data from users.

Simulations have been used to show the performance in a controlled environment and the effect of the amount of RSSI updates on the localisation error. Live tests in non-trivial environments showed that we can achieve room level accuracy and that the localisation can be performed in real time. This is all done locally, i.e. running on a user's device, with respect for privacy and without using any prior information of the environment or device locations.

Although promising, more work is required to increase accuracy in larger environments and to make the algorithm more robust for environment noise caused by walls and other objects. Existing techniques, e.g. map fusing, can alleviate these problems.

***Index Terms*—Smart Homes; Ubiquitous computing; Simultaneous localization and mapping; Wireless sensor networks; Privacy**

I. Introduction

With advances in electronics and computer science, technology has become an indispensable part of our daily lives. A new field, where intelligent systems have not (yet) been fully integrated, is the work and living environment: our homes and offices. Although these spaces are full of intelligent devices, there is often no link or collaboration between intelligent devices and the environment we live in. The current development of the *Internet of Things (IoT)* [1], which attempts to connect devices and, by doing so, creating *smart spaces*, is about to change that.

A key problem (or challenge) within these spaces is indoor localisation: making estimates of users' whereabouts. Without such information, systems are unable to react on the presence of users. This can range from simply turning the lights on when someone enters a room to customising the way devices interact with a specific user.

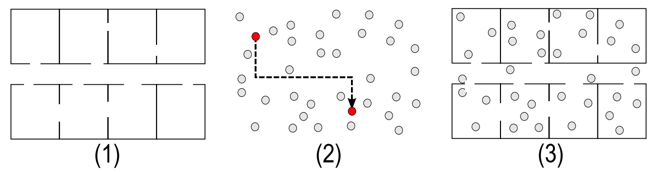


Fig. 1. In a building (1) a set of devices (grey dots) is installed (2) with a user (red dot) that walks around (dashed line). Based on measured RSSI values and motion data from the user we locate devices in the building (3).

Even more important for a system to know exactly where users are, is to know where users are *relative* to the devices the system can control or use to sense the environment. This relation between user and device location is an essential input to these systems. For this we first need a way to find device locations and then locate users relative to those.

By connecting individual devices and sensors we can build systems that, as a whole, interact with a user. For this we require a system that is able to locate users and devices simultaneously. In this paper we propose a new system, called *SLAC*: *Simultaneous Localisation and Configuration*, based on FastSLAM [2], to address these two challenges and combine them into a single estimation problem. With SLAC we aim to simultaneously locate the user and configure a smart space system by finding locations of devices installed inside an indoor environment (see also Figure 1 for an overview).

We will show that this is indeed possible and that localisation of devices can be done very fast while respecting user privacy and using no additional hardware.

A. Potential & Challenges of Smart Spaces

Smart spaces become more effective when there is a smooth interaction between users and devices in a well-mapped environment. For homes, these systems can result in more comfort for residents and increase energy efficiency [3], [4]. For health care it can result in more tailored care, an increase in self-reliance and a decrease in social isolation for patients and elderly [5]–[7]. In a work environment, a smart office can improve work efficiency and job satisfaction [8] and reduce energy consumption [9]. In other words, in the full spectrum of our environment, these smart spaces can have great benefits.

978-1-4673-9948-7/16 $31.00 © 2016 IEEE
DOI 10.1109/IoTDI.2015.19

Not only end users (residents, patients, etc.) but also organisations and other owners of smart spaces can benefit from making their buildings 'smart'. Especially in times where health care costs continue to rise and energy consumption should be minimised, these systems can offer a (partial) solution to these problems.

Regardless of this potential, many challenges still exist which can slow down the adoption of these systems in our environments. Users of smart spaces will, in general, be non-experts that lack the technical skill to install and connect these complex systems. This is not a great problem for large scale applications where specialists can install and maintain systems. However, to make these systems available to general consumers the installation, configuration and management of such systems should be easy. This is one of the main issues that prevents our homes from becoming intelligent: it is too difficult to set up and connect devices to create a collaborative system. Moreover, after these systems have been set up the work is not done. Devices, including sensors, can be moved by users to accommodate for changes in the environment or their condition. And even if installation is carried out by specialists, due to this dynamic nature of our environments, systems should adapt to changes themselves to prevent tedious (re)configuration steps and prevent faults. Direct and explicit interactions with the smart space should be minimised to reduce cognitive load.

B. Indoor Localisation of Devices and Users

In outdoor environments localisation of users is straightforward. Systems such as GPS result in an absolute position of a user: each GPS device can compute its location on a globally consistent coordinate system (e.g. using latitude, longitude and elevation). In indoor environments such methods are often inaccurate or not feasible as signals have difficulties penetrating walls of buildings.

For most systems in smart spaces, an absolute and global coordinate system is not required. The position of a user expressed in latitude and longitude is not particular useful, more interesting is the room or section of a building a user is in. In other words, a system needs to know what the location of a user is *relative* to the environment and the devices therein.

By estimating both the locations of the devices of a system and its users we can compute distances, determine whether two 'things' are close, and track paths of users. This implies more than just the distance between a user and a single device. Relative to the building we want to derive a map on which we can locate users and devices simultaneously. A simple scenario highlights this difference. Consider a user entering a hallway: here we want to turn all the lights in the hallway and not only the lights close to the user.

C. The Need For Privacy-Safe Localisation

Most IoT applications, especially in smart spaces, collect data from users. While this data can be used to solely react instantaneous on user's actions, most practical applications will store data. Storing data makes it possible to perform inference, improve the usability of systems, or give system owners more insights in the use of their system. In such systems, even when data from users is only processed and not stored, we need to tread carefully so as not to infringe on users' privacy. Data collection is often not directly visible for users as it is mostly performed autonomously and in the background.

When movement patterns of users are recorded, a common scenario in indoor localisation, we must protect individual user's privacy or it could potentially hinder the adoption of systems. For end-users it is undesirable that their whereabouts need to be stored remotely in order to use a smart space. Particularly when these systems are deployed in public spaces we must find an equilibrium between storing data and privacy concerns.

Privacy can be improved when the localisation of users and devices is performed locally, on users' devices. This decentralised approach makes it possible to process localisation data without sharing it with a central system. The ownership of location data can then remain in the hands of users.

To perform this local computation we use characteristics that are already available in many smart spaces: signal strength measurements (or *RSSI*) from devices and motion data from smart phones and other portable devices. We combine these two inputs in a system that can locate users and devices, respect individual users' privacy by running locally on a user's device and perform all estimations in real time.

II. Related Work

The vast majority of indoor localisation research focuses on locating a single group of entities: users, robots or devices but not simultaneously. In the case of robots [10] and users [11], *Gaussian Process Latent Variable Models* (*GP-LVM*) have been used as a offline localisation method by creating a signal strength map of WiFi access points. Another common method is fingerprinting where user data is usually sent to a remote server to build a fingerpint database, the latter can be crowdsourced to reduce configuration time [12]. Both these methods usually do not result in very precise estimations, with average localisation errors ranging from $3.79m$for the WiFi GP-LVM approach [11] to $5.88m$ using fingerprint methods [12] . Nonetheless, these results are achieved without prior information of the environment. GP-LVM can also be used for locating devices [10].

Locating devices, in the context of wireless sensor networks, is often done using anchors: devices which have a known position and which can be used to locate the rest of the network. Solutions can assume fully static networks [13]–[15] in which all devices have a fixed position. By moving an anchor the number of devices required for localisation can be reduced [16]. Methods also exist for fully mobile networks, that often focuses on locating swarms of robots [17], [18]. Mobility of devices in these networks can improve localisation performance [19].

A field where localisation is an active topic of research is that of robotics. Robots that are deployed in unknown environments have to simultaneously locate themselves, and map the environment in which they move. This is called the *Simultaneous Localisation and Mapping* problem (or in short *SLAM*).

The estimation of the map and robot position is based on sensor readings and the *controls* (i.e. actions) the robot performed. The combination of both sensor readings and controls is important as the world is nondeterministic and the same control can result in a different outcome. The difficulty of SLAM is mainly caused by a problem which, at first sight, looks like a chicken-and-egg problem: a map is required for estimating a position, but for the mapping an estimate of the robot position is required. SLAM overcomes this problem by estimating both at the same time.

SLAM is often centred around the observation of *landmarks*: distinguishable objects or markers in an environment that a robot can observe. These landmarks can be implicit, like walls, or more explicit such as (sensor) nodes and devices. WiFi access points have, for example, been used as landmarks [20]. When signal strength measurements are used as input it is often required to use a range-only algorithm [21]. Localisation errors for these algorithms in robot applications, due to accurate control data, are often very low, ranging from $2.18m$ using GraphSLAM [20] to $0.46m$ when additional sensors such as camera's are used [22]. Even lower localisation errors can be achieved using inter-device measurements and Sparse Extended Information Filters [23].

III. FastSLAM

Our research builds upon FastSLAM [2] which is an online SLAM algorithm that uses particle filters to do the state estimation. Individual devices, i.e. the landmarks, are represented by *extended Kalman filters* (EKF).

Initially, FastSLAM is defined to solve the *full* SLAM problem: given all sensor readings and robot controls (from $1 : t$), what is the full path and map of the environment? A robot's control is usually its measured movement or the motor commands. The map is represented by a set of landmarks. These landmarks can be anything; in our case they are devices.

The full SLAM problem introduces a conditional independence that the FastSLAM algorithm utilises, using *Rao-Blackwellised Particle filters*, to increase performance: given the robot path, the location of the landmarks are independent of each other and can be estimated separately. See Figure 2 for a visualisation. In order to do this the full SLAM posterior is factorised [24]:

$$\begin{aligned} p(x_{0:t}, m_{1:M} | z_{1:t}, u_{1:t}) &= \text{path posterior} \times \text{map posterior} \\ &= p(x_{0:t}|z_{1:t}, u_{1:t})\ p(m_{1:M}|x_{0:t}, z_{1:t}, u_{1:t}) \quad (1) \\ &= p(x_{0:t}|z_{1:t}, u_{1:t})\ p(m_{1:M}|x_{0:t}, z_{1:t}) \quad (2) \\ &= p(x_{0:t}|z_{1:t}, u_{1:t}) \prod_{i=1}^{M} p(m_i|x_{0:t}, z_{1:t}), \quad (3) \end{aligned}$$

with $x_{0:t}$ the robot's path, $m_{1:M}$ the landmarks, $z_{1:t}$ the sensor measurements and $u_{1:t}$ the controls. Note that the controls $u_{1:t}$

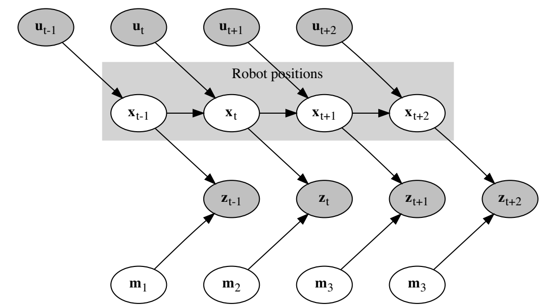


Fig. 2. Bayesian network representation of the FastSLAM algorithm. Grey nodes are observed variables, white nodes are latent. Given the robot path there is no other path between landmarks; i.e. given the path the landmark locations are independent. This independence enables us to estimate each landmark separately.

are omitted in the estimation of the map (Equation 2); given the robot's path the location of the landmarks are independent of the controls. This definition does not incorporate the correspondence (i.e. linking a measurement to a specific landmark) [24]. We assume, for this particular research, that there is no data association problem so the correspondence is not part of the posterior.

By factorising we transform a highly dimensional problem into two separate problems: estimating the path and the landmarks. FastSLAM uses a particle filter to estimate the robot path. The mapping problem, which is easily computable given the robot's path, is factored in to separate problems: one for each landmark. The individual landmark locations can be estimated using a low-dimensional EKF. This is in contrast to other SLAM methods who usually have a joint estimate of all landmarks.

Even though we make an estimate about the whole path of the robot in Equation 3, FastSLAM is primarily used for *online SLAM*: instead of estimating the whole path we only estimate the current position. The definition of the particle filter makes this feasible: the estimate is only dependent on the previous estimate and the whole path is not required.

At time t, a single particle in the FastSLAM algorithm can be described as:

$$Y_t^{[m]} = \Big\langle x_t^{[m]}, \langle \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]} \rangle, \ldots, \langle \mu_{N,t}^{[m]}, \Sigma_{N,t}^{[m]} \rangle \Big\rangle, \quad (4)$$

with $m \in M$ describing the particle index, M the total amount of particles and N the amount of landmarks. $x_t^{[m]}$ describes the estimated robot position and rotation (x, y, θ). Each $\mu_{j,t}^{[m]}, \Sigma_{j,t}^{[m]}$ pair describes the mean and variance of a Gaussian estimate of a single landmark l with $j \in N$.

Calculating the posterior at time t boils down to generating a new particle set Y_t from the previous one Y_{t-1} by executing the three main steps of the FastSLAM algorithm:

1) *Sample (or prediction) step:* Using the robot control u_t a new pose is sampled according to the motion model:

$$x_t^{[m]} \sim p(x_t | x_{t-1}^{[m]}, u_t). \tag{5}$$

The new pose is always computed using the previous estimate from the same particle, $x_{t-1}^{[m]}$.

2) *Landmark update step:* For each observed landmark we update the estimate by updating the mean $\mu_{j,t-1}^{[m]}$ and covariance $\Sigma_{j,t-1}^{[m]}$ using the EKF update function. If an landmark is not observed the estimated position remains unchanged. When a new (previously unseen) landmark is observed we initialise the EKF with the current measurement.

3) *Resampling (or correction) step:* Draw a new set of particles with probabilities proportional to the importance weights. This resampling is required as the measurements are not embedded in the distribution sampled in the sample step. From the sample step, we obtain a distribution given only our motion model. This is not equal to our target distribution which favours positions based on the measurements. By sampling particles based on the weight, which is based on the measurements, we correct for this mismatch.

IV. Signal Strength (RSSI)

Most components in an IoT context communicate through some wireless system such as *ZigBee*, *Bluetooth* or *WiFi*. These communication layers come with a 'free' measure that can be used in localisation: the *Received Signal Strength Indicator* (RSSI). The RSSI value resembles the power of a received radio signal (measured in *dBm*). The higher the RSSI value, the higher the signal strength.

The SLAM approach requires two types of input: measurements or predictions of motion (i.e. the control) to make pose estimations and environment data to resample particles and to build the map (in our case, finding locations of devices). In our application, signal strength (RSSI) measurements are used for the latter.

The rationale behind using RSSI values is that almost all wireless systems report and use this value natively; i.e. no additional sensors are required to measure RSSI values. It can therefore be considered as a free input to a system. Moreover, and this is specifically interesting for localisation, there is a relation between RSSI and distance which can be roughly described using the *Log-distance path loss model* (6) [25], [26].

$$\text{RSSI} = -10n \log_{10}(\frac{d}{d_0}) + A_0, \tag{6}$$

where d describes the relative distance between transceiver and recipient, n the signal propagation exponent and A_0 a referenced RSSI value at d_0. Usually d_0 is taken to be 1 such that A_0 becomes the single strength measured at a distance of 1 meter of the node.

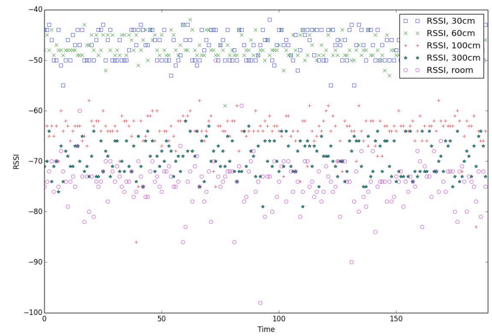


Fig. 3. The received signal strength (RSSI) of a device is influenced by distance, but the amount of noise is substantial. For this figure, a Bluetooth device was set up as a beacon to continuously broadcast its unique identifier. Another device was placed at various distances and acted as a recording device. With a 1 Hz sample rate RSSI values were sampled. For the 'room' case, the transmitter was placed in an adjacent room to show the effect of walls.

A. RSSI Filtering

In an ideal world, the RSSI value is only dependent on the distance between the two devices. In reality, however, RSSI values are heavily influenced by the environment and have, consequently, high levels of noise. It has even been suggested that these noise levels are unacceptable for practical applications [27]. Noise is among others caused by multi-path reflections: signals reflect against objects in the environment (see Figure 3). While the precision of RSSI is limited, and other methods such as sonar are more accurate, we still opted for this method given its availability in consumer devices.

To address the noise problem a (regular) Kalman Filter is applied to filter signal strength measurements. Devices are assumed to be fairly static to simplify the filter; this does not restrict us to devices that not move. The true RSSI value (without noise) is defined as the estimated state. The transition and observation model can then be reduced to

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t = x_{t-1} + \epsilon_t, \tag{7}$$

$$z_t = C_t x_t + \delta_t = x_t + \delta_t, \tag{8}$$

where ϵ_t and δ_t describe Gaussian noise and A_t, B_t and C_t the transition models. A_t and C_t are set to identity matrices as we assume the state is static (i.e. $x_t = x_{t-1}$) and the state is modelled directly (i.e. we assume $x_t = z_t$). Because there is no control, B_t is set to zero. The prediction step of the Kalman filter then becomes:

$$\bar{\mu}_t = \mu_{t-1}, \tag{9}$$

$$\bar{\Sigma}_t = \Sigma_{t-1} + R_t, \tag{10}$$

with R_t as the process noise which is typically set to a small value (e.g. 0.008). The measurement vector is a scalar value set to 1, this gives us the following reduced Kalman gain:

$$K_t = \bar{\Sigma}_t (\bar{\Sigma}_t Q_t)^{-1}. \tag{11}$$

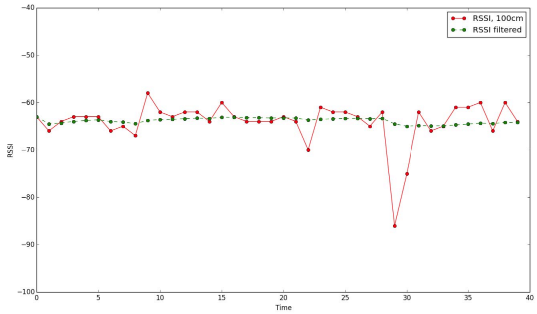


Fig. 4. The effect of a Kalman filter on raw RSSI data sampled from a static device (i.e. no movement on both the receiver or transmitter end). The Kalman filter removes a large part of the noise from the signal.

The measurement noise, Q_t, is set to the variance of the RSSI measurements. The state can then be updated given each new measurement:

$$\mu_t = \bar{\mu}_t + K_t(z_t - \bar{\mu}_t), \tag{12}$$

$$\Sigma_t = \bar{\Sigma}_t - (K_t\bar{\Sigma}_t). \tag{13}$$

The result of the Kalman filter on a sample of raw RSSI data can be seen in Figure 4. The Kalman filter is able to remove a large part of the noise from the data, but as a tradeoff, has to give up a bit of the responsiveness.

V. Motion Measurements

RSSI measurements only give estimates of distance. To perform efficient localisation and creating consistent maps of the environment we need some notion of *where* these measurements took place. In a general SLAM application this is covered by the robot's control: usually its motor commands. In situations with humans we do not have this kind of data. We therefore need to estimate (as opposed to directly derive) motion. For this we use *inertial measurement unit (IMU)* data.

A compass and accelerometer are used to measure user motion. We have to note that these type of sensors don't give the most reliable measurements. Especially compass readings are influenced by indoor magnetic fluctuations. Though, these sensors are present in almost any modern mobile device (including phones, tablets and wearables) and by using them we remove the need for additional and external sensors. This is a trade-off between performance and implementation cost where we favour the latter.

A compass returns the current rotation or heading relative to the global north. Compass data does not require any processing, but additional filtering and smoothing is advised for real world applications. Accelerometers return acceleration in three axes, x, y, z, and need to be processed to make estimations about the distance that is travelled.

Measured acceleration is used as input for a pedometer. The pedometer counts steps which can then be converted to distance. Our pedometer is based on an existing design by [28], [29] and uses a sliding window of the acceleration data to determine whether a step has been made.

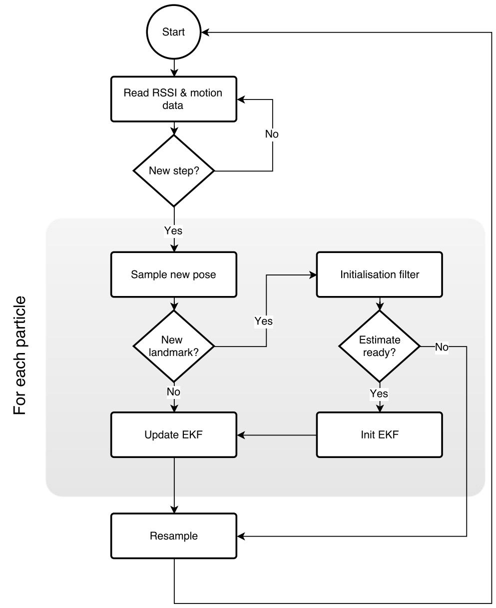


Fig. 5. Visualisation of a single step of the SLAC algorithm. The diagram shows the update step of a single device based on a single observations. In reality, multiple observations and devices can be updated between the sample and resample step.

VI. Simultaneous Localisation and Configuration (SLAC)

With our input defined (the RSSI measurements and motion estimates) we map the SLAM problem to the domain of indoor localisation:

- *Control $\rightarrow$ motion estimates:* The robot's controls, which are used for pose sampling, are replaced by motion estimates of user movement.
- *Landmark $\rightarrow$ device:* As landmarks we use the devices which we try to locate. To be coherent with the SLAM literature we will continue to call them landmarks.
- *2D observations $\rightarrow$ 1D RSSI measurements*: The observations, which are often 2D measurements[1], are replaced by 1D RSSI measurements resulting in a 1D range-only version of FastSLAM [21].

Figure 5 shows an overview of the algorithm.

A. Pose Sampling

The flow and update rate of the SLAC algorithm is controlled by the pedometer: the algorithm is run after a new step has been detected. The step size (the distance a user moves after taking a single step) can be taken as a fixed value or derived from the human's body height. Usually a factor of 0.3 to 0.5 times the body height is used.

[1]When directional range sensors are used angle information of the observations can be derived.

Given the step count[2], the prediction of the step size (r) and the current heading (θ) (taken from the compass) a new pose is sampled for each particle m:

$$\begin{aligned}\bar{\theta} &= \mathcal{N}(\theta, \sigma_\theta),\\ \bar{r} &= \mathcal{N}(r, \sigma_r),\\ x_t^{[m]} &= x_{t-1}^{[m]} + [\bar{r}\cos(\bar{\theta}), \bar{r}\sin(\bar{\theta})],\end{aligned} \quad (14)$$

where σ_θ describes the variance or noise of the compass readings and σ_r the variance of the estimated walking distance.

B. Initialisation Using Particle Filters

Given the sampled pose of each particle we estimate the locations of each landmark (i.e. device) individually. As landmark observations are 1D and signals propagate spherical it is impossible, given a single measurement, to determine the bearing of an observation. This can, for example, lead to mirroring errors. We therefore must make an initial estimate of a landmark location before we can refine it using the default method of FastSLAM.

An often used approach to overcome this initialisation problem is to divide the environment into a grid and use a voting scheme to find the most probable cell in this grid [21], [30]. Applications of this approach usually focus on robots (which have better motion modelling) or on relative small environments. For our indoor localisation we did not want to rasterise the environment so we opted for a different approach using a separate particle filter.

Given a range measurement z with variance σ_z we know that our landmark is somewhere on a circle with a radius equal to this measurement and a bandwidth proportional to σ_z. We then create a new particle filter with N particles which reside on this circle, with our current estimate of the user's position as its centre. The distance and angle (relative to the user) for each particle $i \in N$ are defined by:

$$d_i = \mathcal{N}(z, \sigma_z), \quad (15)$$

$$\theta_i = \frac{2\pi i}{N}. \quad (16)$$

Distance (d_i) and angle (θ_i) are then converted to Cartesian coordinates.

Given enough new measurements this filter can now roughly estimate the device's location. After each new measurement we update our filter by computing the importance weight and subsequently resampling the filter (using low variance resampling). A particle's weight is updated using the probability density function (f) of the normal distribution:

$$w_{i,t} = w_{i,t-1} f(z|d_{i,u}, \sigma_z), \quad (17)$$

where $d_{i,u}$ is the distance between the user and the particle's estimate of the landmark (i.e. the expected measurement). How the particle filter converges to a landmark's location is depicted in Figure 6. When the variance between particles is low enough (given some threshold), we assume that a landmark's location has been found.

C. Refining Using Extended Kalman Filters

After the initialisation filter has converged, we want to further refine the estimate. Our initialisation filter is a separate component that uses the current best user estimate as input. So, to improve the rough initial estimate the estimation must be moved from the global initialisation filter to each individual particle. As it is impractical to update $M \times N$ particle filters (one particle filter per landmark per particle) we use an EKF to estimate a landmark's location (analogous to the original FastSLAM implementation).

The state that our EKF tries to estimate is a 2D position vector; i.e. the x and y coordinates of the landmark. Our observations are however range-only and 1D which results in a slightly different EKF implementation. This particular EKF implementation is based on the implementation of a range-only robot navigation problem [21].

To initialise the EKF we use the estimate from the initialisation filter. The initial covariance matrix of the EKF of a landmark j is defined by the variance of the estimate:

$$\Sigma_{j,0}^{[m]} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}. \quad (18)$$

Given a new measurement z we can update the EKF as follows. For each particle m we start with defining the transition between our state and our expected observation: this is the Euclidean distance between our estimate of the user and our estimate of the landmark:

$$h_m(x_b^{[m]}, y_b^{[m]}) = \sqrt{(x_{u,t}^{[m]} - x_b^{[m]})^2 + (y_{t,u}^{[m]} - y_b^{[m]})^2}. \quad (19)$$

In other words, for any landmark b, the function $h_m(x_b^{[m]}, y_b^{[m]})$ defines the distance between that landmark and the user estimate $[x_{u,t}^{[m]}, y_{u,t}^{[m]}]^T$ of particle m where u resembles the user and t the current time step. As the user estimate is constant, given a specific particle, only the landmark's location is a variable in this function.

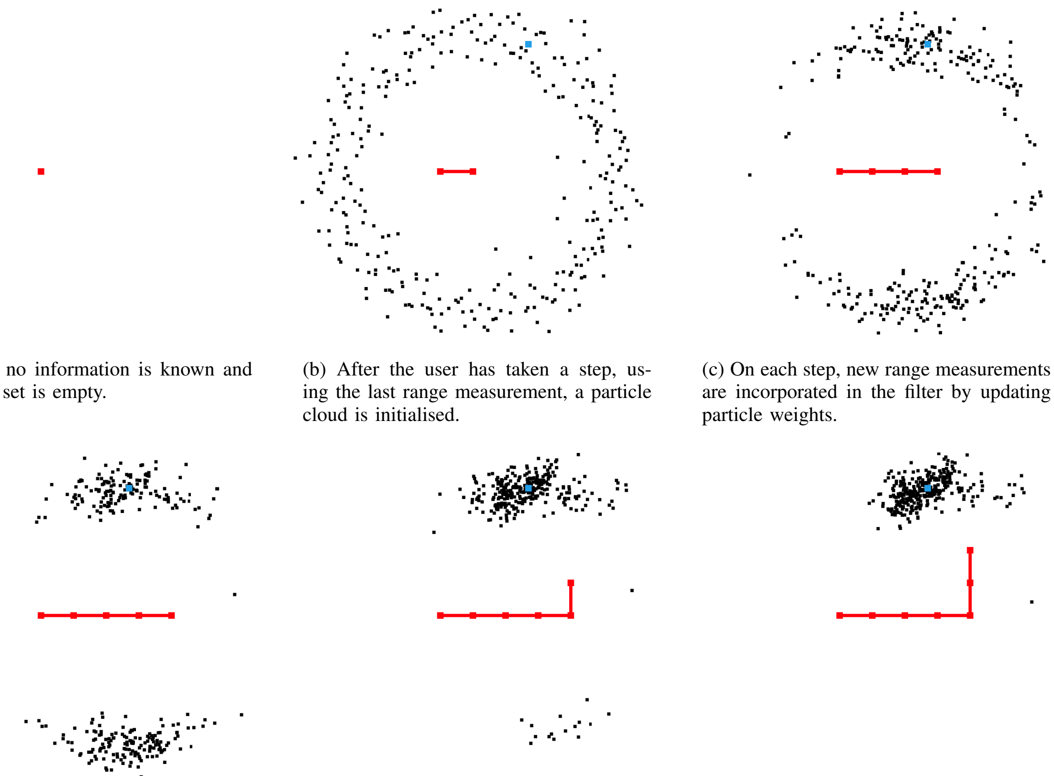
We then calculate the *innovation* which resembles the error from our state to the observation:

$$v = z - h_m. \quad (20)$$

As the transition from our state to observation is nonlinear we have to linearise. We define the Jacobian by computing the partial derivatives of our transition function h_m in both x and y:

$$H = \frac{\partial h_m}{\partial [x_b^{[m]} y_b^{[m]}]} = [\frac{x_{u,t}^{[m]} - x_b^{[m]}}{h_m}, \frac{y_{u,t}^{[m]} - y_b^{[m]}}{h_m}]^T. \quad (21)$$

[2]In our simulations and live tests, the algorithm runs faster than the average time between steps. We can therefore compute a whole update after each step which results in a constant step count of one. However, this is not a requirement and multiple steps can be taken into account, but this can lower the accuracy of the motion sampling.



(a) At start no information is known and the particle set is empty.

(b) After the user has taken a step, using the last range measurement, a particle cloud is initialised.

(c) On each step, new range measurements are incorporated in the filter by updating particle weights.

(d) As RSSI measurements do not contain angle information, multiple solutions can arise when walking in straight lines. This can result in mirroring problems (i.e. two probable solutions).

(e) After a user makes a turn, the particle filter contains enough information to discard one of the solutions.

(f) Given enough movement, the filter eventually converges to a good estimate of the landmark location.

Fig. 6. Initialisation of a new landmark (blue square) is done using a separate particle filter. Given range measurements and movement of a user the initial position of a landmark is estimated. Note that in these visualisations user positions are assumed to be known.

Based on the Jacobian we calculate the Kalman gain. Note that, due to our static motion model for the landmarks, the covariance matrix estimate is not updated (i.e. $\bar{\Sigma}^{[m]}_{b,t} = \Sigma^{[m]}_{b,t}$).

$$\sigma_v = H\Sigma^{[m]}_{b,t}H^T + Q_t \tag{22}$$

$$K = \Sigma^{[m]}_{b,t}H^T\sigma_v^{-1} \tag{23}$$

Given the innovation and the Kalman gain we can update our state and covariance. As our transition model assumes static landmarks we directly update the previous state. The Kalman gain is used as a weighting mechanism.

$$[x^{[m]}_{t+1}, y^{[m]}_{t+1}]^T = [x^{[m]}_t, y^{[m]}_t]^T + Kv \tag{24}$$

$$\Sigma^{[m]}_{t+1} = \Sigma^{[m]}_t - K\sigma_v K^T \tag{25}$$

The last step is updating the importance weight of the particle:

$$w^{[m]}_t = w^{[m]}_{t-1} f(z|h_m, \sigma_z). \tag{26}$$

Updating the previous state and computing the weight completes processing the measurement. This process is performed for each landmark and for each particle.

D. Resampling

After the observation is processed all particles have been updated and contain new importance weights. We can now perform resampling. We utilise *Sequential Importance Resampling (SIR)* to prevent resampling on each step and only perform resampling when there is enough information in the particles. The importance weights are computed and normalised, which are used to calculate the effective number of particles ($\hat{N}_{\text{eff}}$). If the effective number of particles drops below a given threshold we resample.

$$\hat{N}_{\text{eff}} = \frac{1}{\sum_{i=1}^{N}\left(w^{[i]}_k\right)^2} \tag{27}$$

Low variance sampling is used to sample new particles. This type of sampler only requires a single random number

Fig. 7. SLAC application running on both an *iOS* tablet and an *Android* phone.

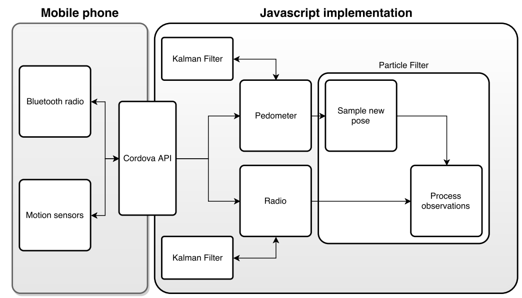


Fig. 8. Visualisation of the Javascript implementation. The Cordova API links the native Bluetooth and motion modules (left) to the Javascript components (right). The data flow is left to right: sensor data flows from the phone, through the API to the particle filter.

to make the selection[3]. The most important advantage of the low variance sampler is its time complexity: $O(M)$ with M the amount of selected particles. In comparison, roulette wheel selection has a complexity of $O(MlogM)$[4].

E. Implementation Details

SLAC has been fully implemented in *Javascript (ECMAScript 2015)*. Javascript was chosen to support a large range of devices on which the algorithm can run; including web browsers and mobile phones. The only requirement, for the current implementation, is access to a radio and IMU data. We chose Bluetooth LE as the communication method for the devices. Landmarks were *iBeacon* compatible Bluetooth radio's. Distance and motion estimates can also be replaced with other methods (e.g. sonar sensors that report distances).

The *Apache Cordova* platform[5] was used to access native APIs of mobile devices. This link between native components and the Javascript implementation is show in Figure 8. Figure 7 shows an example of the application running on both a tablet and a mobile phone. The full source code is available online[6] and is licensed under the *GNU Lesser General Public License v3*.

The SLAC algorithm is an event-based system where the motion data controls the flow. Every time the pedometer detects a new step a single run of the algorithm is performed. All RSSI measurements that were recorded since the last step are used in the update.

VII. Evaluation

Evaluation of the SLAC system started with simulations; this provided the opportunity to repeat the experiment and control the environment. A simulated environment was created with similar dimensions as our real world test bed. The environment was intentionally left empty to asses the performance of the algorithm in an open space environment. On the same coordinates as in the real world, simulated devices were placed. Each device broadcasts messages with signal strengths following the path loss model (with added noise).

[3]Roulette wheel selection, for example, chooses a set of random numbers equal to the amount of particles that needs to be selected.

[4]M random numbers have to be drawn and for each random number a particle has to be selected from the list.

[5]https://cordova.apache.org/

[6]https://github.com/wouterbulten/slacjs

A simulated user walked around in the environment following a fixed path. After each step the algorithm was run. The path is shown in Figure 9. We used a fixed step size of $0.5m$ and always perform 66 steps[7]. Gaussian noise was added to both the step size and the heading.

The frequency of the RSSI updates of each device was varied. As the signal strength is used to make range estimates, the number of received messages can have an effect on the overall performance. The different settings are: $1, 2, 5, 10, 25, 50$ and 100 updates per step. The update frequency is per device. Each setting is simulated 500 times.

The main quantitative measure of performance is the average localisation error: the average distance (in meters) between the device location estimates and their true locations.

A. Online Recordings, Offline Evaluation

Simulating RSSI values and movement has its drawbacks: noise is predictable and there is less interference from the environment. In general it is hard to fully simulate a real environment. In order to asses the performance of the algorithm outside a simulated world, we tested the algorithm inside a real building. Currently our test setup was limited to a single environment.

While SLAC runs online and in real time the data at this location has been recorded and analysed offline at a later stage. The algorithm did however run during the data collection to give feedback about the process. Each recorded data set consists of the raw unprocessed and unfiltered motion data (i.e. acceleration and heading) and RSSI measurements. Each data point has a timestamp which is used to play back measurements in the correct order. This process was repeated several times to get an average performance; this is particular important as the algorithm is a random process: using the same input data twice will result in different outcomes.

[7]66 steps is the average from the live tests.

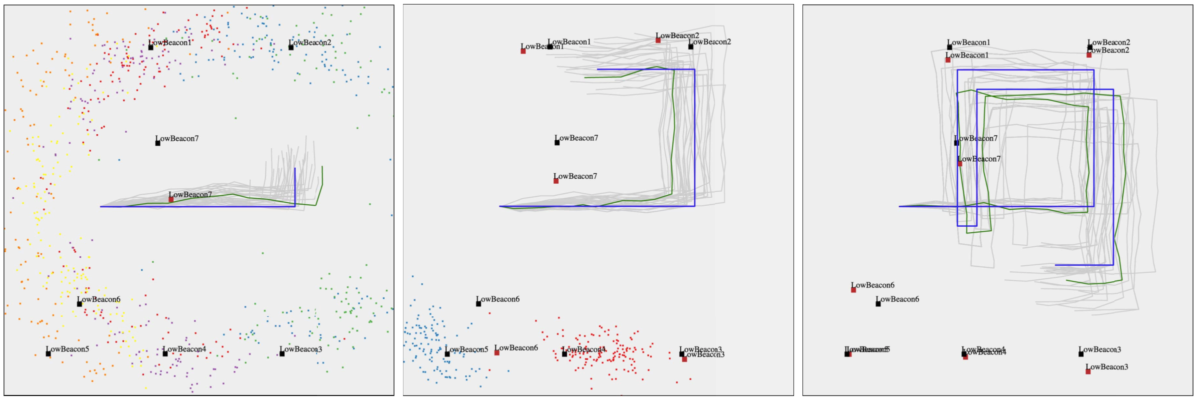


Fig. 9. Examples of a SLAC simulation at various stages. Blue lines shows the ground truth of the user. Each grey line represents a particle's estimate of the user's path. The green line is the current best estimate. Black squares show the ground truth of landmark locations, red squares represent the current best estimate. Coloured small squares at screen one and two represent individual initialisation filters.

Data was collected by a single person, walking roughly the same pattern, carrying an *iPad* on which the algorithm ran. No additional sensors were used, only the ones present in the device. The recorded step count was between 50 and 70 steps. Each data collection run consisted of roughly two minutes of walking.

VIII. Simulation Results

Seven distinct groups of simulations have been run with different number of RSSI updates per step. Overall, the best result were obtained in the simulations with the highest number of RSSI updates per step. This resulted in an average localisation error of $.46m$; i.e. the estimation of landmark positions differed, on average, 46 centimetres from the ground truth. Table I contains an overview of all the simulation runs.

An inverse curve estimation model was used to predict the average localisation error, given noisy movement measurements, based on the number of RSSI updates per step. A significant regression equation was found ($F(1, 3498) = 33868.750, p < .000$), with an R^2 of .906. The average localisation error is equal to $.431 + \frac{4.390}{\text{RSSI}}$ meters with RSSI the number of updates per step.

In the simulations the only difference between devices is their location which has an effect on how the simulated user interacts with it (see Figure 6 for an example of how this can influence the localisation). For each condition a repeated-measure-ANOVA was conducted on the average localisation error using landmark position as the within-subject factor. Values have been converted using a log_{10} transition as for some conditions the localisation error is not normally distributed[8]. Mauchly's Test of Sphericity has been used to detect violations of the sphericity assumption and the Greenhouse-Geisser correction was applied.

For each condition there was a significant effect of landmark position on the average localisation error, see Table II. The size of the effect ranged from very large ($eta^2 > .2$) for conditions with only few RSSI updates to small ($eta^2 < .1$) for conditions with many RSSI updates. In other words, there is a significant effect of landmark position but this effect declines with an increasing number of RSSI updates per step.

[8]Due to low localisation errors the distribution is right skewed.

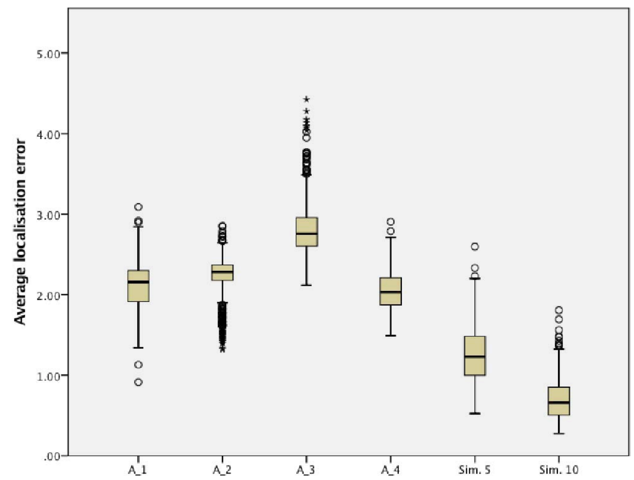


Fig. 10. Showing the average localisation error of the four live test runs (first four) in comparison to the simulations (last two).

IX. Live Test Results

The four live tests resulted in 2000 data points (500 per test). Of these 2000 runs, 108 did not find locations for all devices but had only rough estimates. Based on the step count and the number of RSSI updates we derived that the average number of RSSI updates per algorithm step lays between 7 and 10. The results of the live runs will therefore be compared with the *5* and *10 RSSI updates* simulations. An overview of the individual runs is shown in Figure 10.

An independent-samples t-test was conducted to compare the average localisation error in the live tests with the *5 RSSI updates* simulations. There was a significant difference

TABLE I
TEST RESULTS. THE 'COMPLETE' COLUMN OF THE TABLE DESCRIBES THE PERCENTAGE OF RUNS THAT FOUND AN ESTIMATE FOR ALL DEVICES.

Run	**localisation error (*m*)**	**sd of error**	**Complete (%)**
Sim., 1 RSSI	4.49	.57	75
Sim., 2 RSSI	3.35	.54	99
Sim., 5 RSSI	1.25	.34	100
Sim., 10 RSSI	.69	.25	100
Sim., 25 RSSI	.50	.20	100
Sim., 50 RSSI	.48	.18	100
Sim., 100 RSSI	.46	.19	100
Live A 1	2.10	.31	95
Live A 2	2.20	.30	100
Live A 3	2.84	.36	92
Live A 4	2.04	.25	100

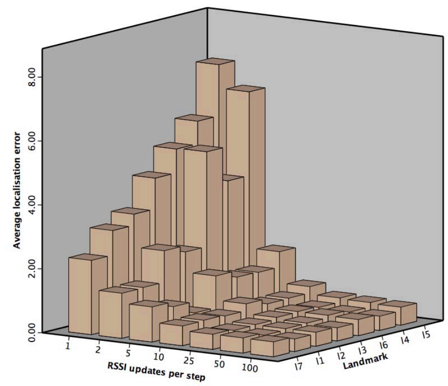


Fig. 11. Localisation error for individual devices (landmarks) for simulations with different RSSI frequencies. Both the position and the number of updates have an effect on the error.

TABLE II
REPEATED-MEASURE-ANOVAS USING LANDMARK POSITION AS THE WITHIN-SUBJECT FACTOR.

RSSI	**Effect**		**Partial eta^2**
1	$F(3.584, 1369.246) = 386.167$	$p = .000$	.503
2	$F(4.473, 2227.97) = 1152.524$	$p = .000$	.698
5	$F(5.475, 2227.97) = 198.102$	$p = .000$	.284
10	$F(5.529, 2758.900) = 33.274$	$p = .000$	.063
25	$F(5.060, 2525.068) = 27.325$	$p = .000$	.052
50	$F(5.087, 2538.630) = 34.545$	$p = .000$	.065
100	$F(5.170, 2580.006) = 34.544$	$p = .000$	.065

($t(2498) = -59.527, p = .000$) in average localisation error for the live runs ($M = 2.29, sd = .44$) and simulations ($M = 1.25, sd = .34$).

A second independent-samples t-test was conducted to compare the average localisation error in the live tests with the *10 RSSI updates* simulations. There was a significant difference ($t(2498) = -78.524, p = .000$) in average localisation error for the live runs ($M = 2.29, sd = .44$) and simulations ($M = .69, sd = .25$).

On average the localisation error is 2.3 m. The real life conditions add between 1.04 and $1.6m$ to the localisation error when comparing with the simulations.

X. DISCUSSION

In this research we set out to explore the possibility of designing a fully autonomous self-localisation system for a network of devices by utilising the users of the space that the network is deployed in. Our main goal was to perform this localisation without any prior information about the environment or the devices therein while retaining user privacy.

We have shown, through simulations, that within controlled environments (i.e. Gaussian noise, no obstacles), we can achieve an average localisation error below half a meter. This means that, after running the algorithm, the estimate of a device's location will, on average, be within 50 cm from its actual position. Estimates are relative to the user's starting orientation, but are correctly scaled. The only input that is used are signal strength measurements from the devices and motion data from a user. No information about the structure of the room or the position of the device is required.

These simulations show the maximum attainable performance; they were conducted in controlled environments using Gaussian noise and a high update frequency of the devices. When the update frequency of devices is lowered, to a level similar to our live tests, the average localisation error increased and resulted in an average error of .69 to 1.25 m.

After simulations the algorithm was tested in a real world environment of similar dimensions with the same number of devices. Our environment was, however, not a clear open space as our simulation environment, but contained many sources of noise, including objects, walls and people walking around. This was deliberate as these noisy environments are exactly the target platform for this technique. All combined, our live tests showed a localisation error of 2.3 m, averaged over all devices, after letting a single user walk around a fixed route (roughly 60 steps). These results suggest that we can achieve room level accuracy; i.e. detecting in which room a user or device is, and roughly know where in the room. We expect that longer traces result in lower localisation errors.

Our simulations suggest that the number of RSSI updates is the largest contributor to the localisation performance. This follows directly from our system: given more information the Kalman filter, responsible for filtering the raw RSSI signal, will be able to give a less noisy estimate. These distance estimates are vital in updating device positions and weighing particles.

The number of RSSI updates is, however, not the only factor. The live test environment contained objects, walls and humans who walked around. Each of these add a factor of unpredictable noise to the RSSI measurements. The path loss model does not account for these obstructions which can result in incorrect distance estimates. Improving these distance estimates seems to be the most promising method of lowering localisation errors.

Fig. 12. Pictures of our live testbed. Seven landmarks were placed in wall sockets around the room. None of the furniture or objects were removed before testing. The room's dimensions are around $10 \times 10m$.

A. Privacy, Online & No Prior Information

While localisation performance on its own is an important metric it was not the motivation behind this project. Three additional qualitative requirements guided many of the design choices: the localisation should be online, privacy must be assured and no prior information should be used.

Rao-Blackwellized filters and an efficient implementation of the algorithm enable us to predict a new state based only on the previous one. This results in an very efficient method that can do estimation in real time; in contrast to offline methods which only give these estimates after processing all data. As a tradeoff we lose a bit of localisation accuracy; the online algorithm will never be able to outperform offline methods who use the full trace.

By running the full algorithm locally on a user's device, and without communicating with the devices it tries to locate, we improve user privacy. The only input, from sources outside the user's own device, are RSSI measurements. No information about the user is sent to external devices. We have to note that by using a wireless device inside a building a user can still be tracked by external measures. Nonetheless, for the localisation itself, no outgoing communication is required.

No additional hardware or information about the environment is required due to the combination of device data and user motion. The only prior information the SLAC algorithm needs is an estimate of the step size of the user (although a general average could be used for this) and the signal strength of devices at a 1 meter distance. This last value is often part of the broadcast message.

B. Improving Performance With Map Fusing

The system, as proposed in this research, runs completely on a user's device with the practical side effect of being privacy friendly. In large environments users will however not always traverse through the full building. Moreover, using the trace of a single user it can be hard to get good estimates of device locations. Assuming we do not want to force a single user to walk around the whole building for a longer period of time we need to add additional measures.

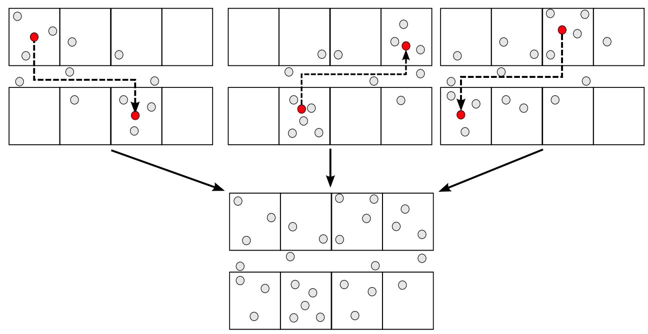


Fig. 13. Overview of the idea of map fusing. Three different users (top, red dots) walk around (dashed lines) inside the same building but their individual runs only result in partial estimates of the environment (devices are depicted as grey dots, walls are black lines). By fusing these estimates together a more complete map of the environment can be created (bottom).

This can be remedied by combining data from multiple users: the process of *map fusing*. With map fusing we attempt to combine individual user's maps into a single globally consistent map. See Figure 13 for an overview. Individual errors, such as mirroring errors, can potentially be removed using these kinds of methods.

When individual user's maps are collected, in order to fuse them together later, there is a risk of lowering user privacy when their path is required to construct the map. As we are primarily interested in device positions, the specific path a user took is less relevant. The factorised approach we used in this research allows us to only share predicted device locations and facilitates privacy-aware map fusing.

The individual device estimates are independent of each other and also independent of the user's path given we can 'ground' the user's individual map to some global frame. The pedometer and our path loss model give 'real world' distances, thus we only need to rotate maps and find the starting positions of the users to align them. A user's initial position, which can for example be an entrance to a building, and the individual EKFs (for each device) are enough to perform the map fusing. Thus, we do not share when or where the user has seen the devices, only their estimates. It remains a task to find a correct estimate of a user's initial position.

Map fusing offers an interesting follow-up research to the SLAC algorithm; it offers the possibility to distribute the mapping of the environment to multiple users while still maintaining user's privacy. Moreover, by fusing individual EKF's the localisation error can be decreased.

C. Conclusion

In this research the simultaneous localisation and mapping problem from robotics was translated to indoor localisation of smart spaces with a focus on locating (smart) devices. The traditional SLAM approach was adapted to use 1D signal strength measurements as input. A motion model, using accelerometer and compass data as input, was used to make estimations of user movement. The full algorithm runs online, with the full update function computed in real time and does not require historical data to run.

Live tests, in a non-trivial environment and with consumer devices, showed that room level accuracy is possible and that localisation of devices can be done very fast. This is all done locally, i.e. running on users' devices, with respect for user privacy and without using prior information of the environment.

More work is required to increase accuracy in large environments and to make the algorithm more robust to environment noise. Existing techniques can alleviate these problems, e.g. by implementing map fusing and letting users work together.

ACKNOWLEDGMENT

This work has been supported by the Proheal project (n. 12007) funded by the Information Technology for European Advancement (ITEA2).

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," *Proc. of 8th National Conference on Artificial Intelligence/14th Conference on Innovative Applications of Artificial Intelligence*, vol. 68, no. 2, pp. 593–598, 2002.

[3] D. D.-M. Han and J. J.-H. Lim, "Design and implementation of smart home energy management systems based on zigbee," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 3, pp. 1417–1425, Aug. 2010.

[4] N. C. Batista, R. Melício, J. C. O. Matias, and J. P. S. Catalão, "Photovoltaic and wind energy systems monitoring and building/home energy management using ZigBee devices within a smart grid," *Energy*, vol. 49, no. 1, pp. 306–315, 2013.

[5] M. Chan, D. Estève, C. Escriba, and E. Campo, "A review of smart homes- present state and future challenges." *Computer methods and programs in biomedicine*, vol. 91, no. 1, pp. 55–81, Jul. 2008.

[6] J. A. Stankovic, Q. Cao, T. Doan, L. Fang, Z. He, R. Kiran, S. Lin, S. Son, R. Stoleru, and A. Wood, "Wireless sensor networks for in-home healthcare: Potential and challenges," in *High confidence medical device software and systems (HCMDSS) workshop*, 2005, pp. 2–3.

[7] M. Skubic, G. Alexander, M. Popescu, M. Rantz, and J. Keller, "A smart home application to eldercare: current status and lessons learned." *Technology and health care : official journal of the European Society for Engineering and Medicine*, vol. 17, no. 3, pp. 183–201, Jan. 2009.

[8] C. Röcker, "Services and applications for smart office environments-a survey of state-of-the-art usage scenarios," in *Proceedings of the International Conference on Computer and Information Technology (ICCIT 2010), Cape Town, South Africa*, 2010, pp. 1173–1189.

[9] M. Choi, W.-k. Park, and I. Lee, "Smart Office Energy Management System Using Bluetooth Low Energy Based Beacons and a Mobile App," in *Consumer Electronics (ICCE), 2015 IEEE International Conference on*, 2015, pp. 501–502.

[10] G. A. Hollinger and J. A. Djugash, "Tracking a Moving Target in Cluttered Environments with Ranging Radios," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 2008, pp. 1430–1435.

[11] B. Ferris, D. Fox, and N. Lawrence, "WiFi-SLAM Using Gaussian Process Latent Variable Models." in *IJCAI*, 2007, pp. 2480–2485.

[12] Z. Yang, C. Wu, and Y. Liu, "Locating in fingerprint space: wireless indoor localization with little human intervention," in *Proceedings of the 18th annual international conference on Mobile computing and networking*, 2012, pp. 269–280.

[13] P. Ekberg and E. C.-H. Ngai, "A distributed Swarm-Intelligent Localization for sensor networks with mobile nodes," in *2011 7th International Wireless Communications and Mobile Computing Conference*. Ieee, Jul. 2011, pp. 83–88.

[14] L. Li and T. Kunz, "Localization Applying An Efficient Neural Network Mapping," in *Proceedings of the 1st International ICST Conference on Autonomic Computing and Communication Systems*. Icst, 2007.

[15] P.-J. Chuang and C.-P. Wu, "An Effective PSO-Based Node Localization Scheme for Wireless Sensor Networks," in *2008 Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies*. Ieee, 2008, pp. 187–194.

[16] K. Sreenath, F. Lewis, and D. Popa, "Simultaneous adaptive localization of a wireless sensor network," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 11, no. 2, pp. 14–28, 2007.

[17] J. Sheu, W. Hu, and J. Lin, "Distributed Localization Scheme for Mobile Sensor Networks," *Mobile Computing, IEEE Transactions on*, vol. 9, no. 4, pp. 516–526, 2010.

[18] L. Hu and D. Evans, "Localization for mobile sensor networks," in *Proceedings of the 10th annual international conference on Mobile computing and networking - MobiCom '04*. New York, New York, USA: ACM Press, 2004, pp. 45–57.

[19] A. Baggio and K. Langendoen, "Monte Carlo localization for mobile wireless sensor networks," *Ad Hoc Networks*, vol. 6, no. 5, pp. 718–733, Jul. 2008.

[20] J. Huang, D. Millman, M. Quigley, D. Stavens, S. Thrun, and A. Aggarwal, "Efficient, generalized indoor WiFi GraphSLAM," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. Ieee, May 2011, pp. 1038–1043.

[21] D. Sun, A. Kleiner, and T. M. Wendt, "Multi-robot range-only SLAM by active sensor nodes for urban search and rescue," in *Robocup 2008: Robot Soccer World Cup XII*, vol. 5399, 2009, pp. 318–330.

[22] E. Menegatti, M. Danieletto, M. Mina, a. Pretto, a. Bardella, S. Zanconato, P. Zanuttigh, and a. Zanella, "Autonomous discovery, localization and recognition of smart objects through WSN and image features," *2010 IEEE Globecom Workshops*, pp. 1653–1657, Dec. 2010.

[23] Torres-González, J. A and Dios, and A. Ollero, "Efficient Robot-Sensor Network Distributed SEIF Range-Only SLAM," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 2014, pp. 1319–1326.

[24] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. the MIT Press, 2005.

[25] S. Y. Seidel and T. S. Rappaport, "914 mhz path loss prediction model for indoor wireless communications in multi-floored building," *IEEE Transactions on Antennas and Propagation*, vol. 40, no. 1, pp. 207–217, 1992.

[26] A. Patri and S. P. Rath, "Elimination of Gaussian noise using entropy function for a RSSI based localization," in *2013 IEEE Second International Conference on Image Information Processing (ICIIP-2013)*. Ieee, Dec. 2013, pp. 690–694.

[27] Q. Dong and W. Dargie, "Evaluation of the reliability of RSSI for indoor localization," in *2012 International Conference on Wireless Communications in Underground and Confined Areas, ICWCUCA 2012*, 2012.

[28] N. Zhao, "Full-featured pedometer design realized with 3-Axis digital accelerometer," *Analog Dialogue*, vol. 44, no. 6, 2010.

[29] S. Ménigot, "Pedometer in HTML5 for Firefox OS and Firefox for Android," 2014.

[30] E. Olson, J. J. Leonard, and S. Teller, "Robust range-only beacon localization," *IEEE Journal of Oceanic Engineering*, vol. 31, no. 4, pp. 949–958, 2006.