2014 International Conference on Indoor Positioning and Indoor Navigation, 27^{th}-30^{th} October 2014

Generating indoor maps by crowdsourcing positioning data from smartphones

Parijat Mazumdar
Department of Electrical Engineering
Indian Institute of Technology Delhi
Delhi, India
Email: mazumdarparijat@gmail.com

Vinay J. Ribeiro
Department of Computer Science & Engg.
Indian Institute of Technology Delhi
Delhi, India
Email: vinay@cse.iitd.ernet.in

Saurabh Tewari
CSR Technologies India Pvt. Ltd.
Noida, India
Email: Saurabh.Tewari@csr.com

***Abstract*—Indoor maps are highly essential for indoor positioning and location-based services. Applications providing navigation support to users are rendered useless without a map of the vicinity being available. Presently, floorplans of public locations are collected and maintained by designated organizations using methods that require excessive manual intervention. This process of creating a database of indoor maps is neither efficient nor scalable to the practically infinite number of public indoor places around the world.**

In this paper, we present a crowdsourcing algorithm to automatically create floorplans of buildings with zero prior information. The algorithm leverages the positioning data shared by pedestrians using smartphone-based navigation systems in the building. It expects only position fixes and associated uncertainties from the navigation systems and does not depend on any particular navigation algorithm. The available positioning data in a completely unknown building is essentially PDR-based and is known to be prone to high amounts of accumulated error primarily due to the lack of reliable error resetting techniques. The presented algorithm takes into account the possibility of such highly erroneous motion traces of pedestrians while trying to generate map as accurately as possible. As an added merit, the algorithm does not depend on the availability of Wi-Fi access points.

I. Introduction

Indoor positioning and navigation has been a hot topic of research for the past few years. Researchers have developed innovative systems and algorithms like NavShoe [1], FootSLAM [2] etc. to make indoor navigation a reality in known as well as unknown environments. However, the dependence of these methods on specialized hardware limit their use. The ubiquitous modern day smartphones, equipped with Wi-Fi receiver and inertial sensors, have changed the scenario altogether. Wi-Fi fingerprinting based methods like the Horus [3] and with PDR based techniques like UPTIME [4] can localize pedestrians in indoor locations with reasonable accuracy using just smartphone and reasonable additional infrastructure like Wi-Fi access points. These methods have been widely accepted commercially and as a result, in a few places, the general public is already reaping the benefits of indoor navigation via easy-to-use smartphone applications. But, despite these advancements, indoor navigation is not as widespread as outdoor navigation using GPS and GNSS. We believe that a primary reason for this is the reduced availability of indoor maps.

Indoor maps are a prerequisite for many popular indoor positioning solutions like the Horus [3]. Even algorithms that do not directly depend on indoor maps, benefit from their availability. Applying map-matching techniques increases the accuracy and consistency of almost any PDR-based solution like the UPTIME [4]. More importantly, indoor maps are a neccessity for the end users of navigation solutions. For assistance in navigation, pedestrians require their present location relative to the structure and positioning of hallways and rooms in the building. In such a scenario, only estimating the latitude and longitude of their current position is useless to them. This positioning data makes sense to them only when supplied with a building floorplan.

Despite the immense importance of the availability of indoor maps, the database of floorplans available with any location-based service provider like SkyHook [5] is limited. This is because the present process of acquiring indoor maps involves intensive manual intervention. Given the vast number of indoor locations worldwide and the fast rate at which new public buildings like shopping malls are being constructed, manually acquiring floorplans is in itself not scalable. The situation is made worse by the fact that original floorplans become outdated with time as the buildings are renovated or their internal structures are altered. There is a need to automate this process of map collecting and maintanence.

In this paper, we present a server-side algorithm which automatically creates and maintains a database of indoor locations, making use of the positioning data shared by pedestrians using their indoor navigation applications in their smartphones. The positioning data required by the mapping algorithm is comprised of the position fix (i.e. latitude and longitude) and its associated uncertainty which can be in one of many standard forms like lat-long deviation, radius of 50% (or 90%) confidence region etc. In our opinion, these requirements of positioning data do not limit the viability of our algorithm because most navigation solutions leveraging smartphone's inertial sensors calculate these values inherently. The mapping algorithm is a truly crowdsourcing one in the sense that it works with the shared positioning data irrespective of the positioning solution used to generate the data. Also, unlike SLAM techniques which depend on looping trajectory of pedestrians, our algorithm expects zero additional effort by

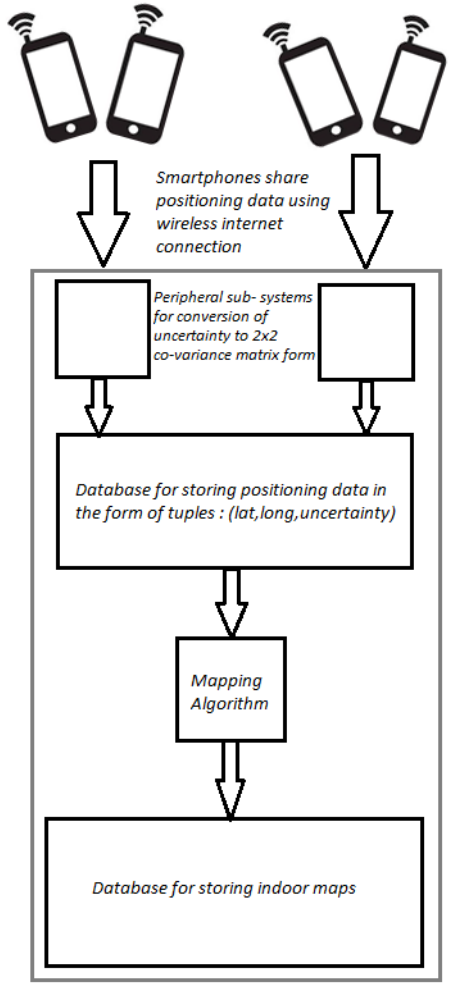


Fig. 1. System Overview

users.

Not controlling the positioning method or the quality of off-the-shelf smartphones used presents the challenge of using a mixture of accurate as well as highly erroneous position fixes in generating indoor maps as accurately as possible. We introduce the concept of *spatial density histograms using varied Gaussian kernels* as a non-parametric statistical way to address this problem.

The rest of the paper is organized as follows. We start with an overview of our algorithm, followed by a discussion on the mathematical model used. Then we lay down our algorithm in specific terms and present the results of our experimentation. Finally, we review related work and conclude our paper.

II. System Overview

In this section, we formulate the indoor mapping problem. We also describe the complete system (see Fig. 1) of which the mapping algorithm, discussed in the rest of the paper, is a central part.

A complete indoor map consists of two important types of information. First, it specifies the position of the walls shaping hallways and rooms along with the position of rarely moved obstacles like tables, decorative installations in shopping malls etc. Second, it specifies the defining labels associated with the various parts segregated by the postioning of the obstacles. For example, in a shopping mall, the labels could be the names of various shops. The work presented in this paper deals with estimating only the former of the two types of information. Once the positioning of walls and other obstacles are estimated, the defining labels can be set through human contribution and feedback or by using prior work like the SurroundSense [6].

The system we propose is a data collection and processing server which maintains a database of indoor maps. Any positioning and navigation application can send positioning data of the user over an Internet connection. Specifically, the server accepts the tuple (latitude, longitude, uncertainty) as positioning data. The mapping algorithm used by the system represents each position fix as a normal distribution with latitude and longitude of the position fix as mean and the uncertainty as the co-variance matrix. Hence it is necessary to represent the uncertainty as a 2x2 co-variance matrix. Some positioning systems like those using Kalman filter internally calculate the covariance matrix denoting uncertainty, but other systems may instead measure uncertainty in terms of latitute and longitude deviation or radius of 50% confidence region. We argue that all standard forms of representing uncertainty can be converted to the co-variance matrix form with inherent approximations. For example, when latitude and longitude deviation form is used, the distributions along latitude and longitude can be considered independent and the co-variance matrix entries can be calculated accordingly. These conversions are inexpensive and can be done by peripheral sub-systems in our system, each of which converts one of the finitely many standard forms of uncertainty to the co-variance matrix form. The mapping algorithm is run periodically over the collected data to generate or modify maps and store them in the database. The positioning data is stored in the system in a time-bound manner, i.e., each data point is stored in the system only for a certain time after which it is discarded. This helps in keeping the maps in the database up-to-date.

At this point, it is worth mentioning that the system can be easily extended to store Wi-Fi RSSI values as well which will enable us to associate Wi-Fi fingerprints with the generated map. This inturn can be useful for a variety of Wi-Fi fingerprint-based positioning applications. Our work presented in this paper, however, focuses only on generating building floorplans from crowd-collected position fixes.

III. Mathematical Problem Formulation and Solution

In this section, we formalize the mapping problem and depict it as a statistical problem. We also introduce *Spatial density histograms using varied Gaussian kernels* as a non-parametric way to tackle the statistical problem, thus generating a probabilistic version of the indoor map.

A. Problem Formulation

The task is to use a collection of uncertain position fixes of pedestrians to generate the map of a building. This problem statement can be defined, in a probabilistic setting, as the estimation of map given all the position fixes of pedestrians.

$$pdf(\mathcal{M}|f_1, f_2...f_n) \tag{1}$$

where $\mathcal{M}$ denotes the map random variable, and each f_i denotes an uncertain position fix modelled as a Gaussian distribution (explained below). $\mathcal{M}$ is a random variable for various possible states of the map where each map state is a specific way to distributing the vacant areas and the blocked areas over a 2-D space. Therefore, we can say that $\mathcal{M}$ captures the various ways of distributing vacant areas over a 2-D space. The i^{th} position fix in the input collection of positioning data is given by the tuple (lat_i,$long_i$,Σ_i), where lat_i is its latitude, $long_i$ is its longitude and Σ_i is its co-variance matrix denoting the uncertainty of the position fix observed. The erroneous position fix is mathematically represented in our problem formulation as an independent bivariate Gaussian distribution, denoted by f_i, with (lat_i,$long_i$) as mean and Σ_i as the co-variance matrix.

$$f_i(lat, long) = \mathcal{N}([lat, long]|\mu = [lat_i, long_i], \sigma = \Sigma_i) \tag{2}$$

where lat and $long$ are the latitude and longitude of the location at which we want to evaluate the value of f_i; $\mathcal{N}$ stands for normal distribution. Since Σ_i is the uncertainty of the position fix observed, we can say that, f_i represents the *pdf* of the actual position of a person given his observed position fix.

Keeping in mind the above inferences, we conclude that the probability expression in (1) represents the probability density function of vacant locations over a continuous 2-D space, given the finite observations of vacant locations (i.e. f_i's). The use of f_i as an observation of vacant location is a simple probabilistic extension of the following conjecture which intuitively applies if position fixes are not erroneous : the location associated with a position fix of a pedestrian is always vacant. In our proposed method, we estimate the bivariate *pdf*, in (1), in a non-parametric way using spatial density histograms. The concept of our spatial density histograms derives heavily from the statistical technique of Kernel Density Estimation (KDE) [7], widely used in computer vision and signal processing. We briefly discuss the essentials of KDE in the next part and then subsequently describe our spatial density histograms.

B. A primer of Kernel Density Estimation

Kernel Density Estimation (KDE) is used to estimate the probability density function of a random variable in a non-parametric way. Given a finite number of independent and identically distributed samples, $(x_1, x_2 \dots x_n)$, drawn from an unknown distribution, the *pdf* of the distribution is estimated using the following equation :

$$pdf(\underline{x}) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{||\underline{x} - \underline{x}_i||}{h}) \tag{3}$$

In the above equation, $K(\bullet)$ is called the kernel function and is controlled by the parameter h called the kernel bandwidth; $||.||$ denotes the Euclidean norm of the vector. The most popular choice of kernel function is the normal kernel described by the following equation :

$$K(d; h) \propto e^{-\frac{d^2}{2h^2}} \tag{4}$$

An important advantage of KDE is that it can asymptotically achieve estimation optimality for a distribution without making any assumptions about it. In other words, as the number of observed samples increases, the *pdf* estimate converges to the true density of ANY distribution [8].

$$\int |\tilde{p}(\underline{x}) - p(\underline{x})| d\underline{x} \to 0 \; as \; N \to \infty \tag{5}$$

where $\tilde{p}$ is the actual *pdf*; p is the estimated *pdf*; N is the number of input samples and $|.|$ is the standard modulus operator for real numbers. This convergence property makes KDE, in our opinion, an ideal technique for crowdsourcing indoor maps. However, we do not directly use KDE for our map estimation problem primarily for one reason. In KDE, the contribution of all training points are identical in nature because the same kernel function is applied to all the points. This is not a suitable property in case of our mapping problem where some position fixes are quite accurate while others can be highly erroneous (as demonstrated by the data collected for our experiment see figure 6b). An ideal solution, hence, should give higher weightage to the more certain position fixes and discount the contribution of highly uncertain points. Our method of *spatial density histogram using varied Gaussian kernels* inherently incorporates this differential weighting strategy.

C. Spatial density histograms using varied Gaussian kernels

The transition from KDE to spatial density histograms comprises of two changes.

1) Discretization of continuous space: We discretize the continuous 2-D space over which the state of map random variable $\mathcal{M}$ is to be estimated by choosing a grid representation of the map. We represent the map as a grid of square tiles of fixed dimensions (δ x δ), each of which is either *vacant* or *occupied*. The vacant tiles are available for pedestrians' use while the occupied ones are blocked by obstacles to human motion (e.g. walls, tables). Each of these tiles, much like individual pieces of a jigsaw puzzle, when put together generate the entire building floorplan. The density estimate within each tile, T_{ij} (i.e. the map tile in the i^{th} row and j^{th} column of the grid), in the map grid is considered to be uniform and same as the density estimate at it's center point, $\underline{c}_{ij}$. In other words, the tiles are 2-D histogram bins whose values are the accumulated density at their center points ($\underline{c}_{ij}$s).

2) Varied Gaussian kernels: We use f_is,described in subsection III-A, for calculating density contributions of position fixes. Each f_i is essentially the output of the corresponding position fix applied to a Gaussian kernel whose kernel bandwidth is proportional to the uncertainty of the position fix. The use

of Gaussian kernels with customized bandwidths makes the contributions of the various position fixes non-identical and dependent on the certainty of the position fixes. Position fixes with high uncertainties have spread-out or flattened f_is. As a result, these position fixes contribute small values to a large area of the map. Such almost-uniform, global contributions are incapable of changing the shape of the *pdf* of the map. On the contrary, highly certain position fixes have sharp f_is and hence have high local contributions to the *pdf*. These sharp, local contributions have significant effect on the shape of the map *pdf*.

In summary, we estimate the density at each tile in the map grid using the following equation:

$$pdf(T_{ij}) \propto \sum_{i=1}^{n} f_i(\underline{c}_{ij}) \tag{6}$$

The density estimates at the map tiles correspond to the likelihood of the tiles being vacant and together form the *pdf* of the indoor map.

The statistical significance of our method is that it estimates the likelihood of a map location being vacant using the *expected number of pedestrians* who have walked over that location, calculated using the erroneous position fix readings. To substantiate this claim, we show the equivalence between the density estimate at a map tile (calculated using (6)) and the expected number of pedestrians who have walked over it. Remember from equation (2) that each f_i represents the *pdf* of the actual position of a pedestrian given his position fix reading. Therefore, the expected number of persons who have been over a particular tile of dimensions $\delta \times \delta$ is given by:

$$E(p_{ij}) = \sum_{i=1}^{n} \int_{X_{c_{ij}}-\delta/2}^{X_{c_{ij}}+\delta/2} \int_{Y_{c_{ij}}-\delta/2}^{Y_{c_{ij}}+\delta/2} f_i(x,y)\, dx\, dy \tag{7}$$

where $X_{c_{ij}}$ and $Y_{c_{ij}}$ are the X-coordinate and Y-coordinate of the tile center $\underline{c}_{ij}$. For δ much less than the variance of f_i, the above equation can be approximated as :

$$E(p_{ij}) = \sum_{i=1}^{n} \delta^2 f_i(X_{c_{ij}}, Y_{c_{ij}}) \propto \sum_{i=1}^{n} f_i(\underline{c}_{ij}) \tag{8}$$

Comparing equation (8) with equation (7), we see that the expected number of people who have walked over tile T_{ij} is proportional to the density estimate at the same tile, thus proving our claim of equivalence between expected number of pedestrians and density estimate.

IV. The Algorithm

Our algorithm can be broadly divided into two parts : generating the probabilistic map using spatial density histograms discussed above and using the probabilistic map thus generated to mark the positions of walls and other obstacles. In this section, we describe both the parts in order.

A. Generating probabilistic maps

The algorithm for generating a probabilistic map is based on the concept of *spatial density histograms with varied kernels* discussed in Section III-C. We take up the various aspects of the algorithm separately. The complete algorithm is summarized in Algorithm 1.

1) Forming a grid representaion: We ideally want to represent the map as a grid of square cells of fixed dimensions as discussed in III-C1. But for computational efficiency and representational ease, we approximate the sets of parallel horizontal and vertical lines, which demarcate the individual squares, with latitudes and longitudes separated by fixed intervals. Within the limited spread of almost all indoor locations, the latitudes and longitudes form two sets of almost parallel lines (see Fig. 2). The slight skewness imparted to the squares as a result of this approximation has negligible effect. The interval size between latitudes (and longitudes) is a design decision and depends on the desired size of individual square tiles. Forming the grid is the first step of our algorithm which can be done by identifying the demarcating latitudes and longitudes. Since the interval between damarcating latitudes and longitudes is fixed (say Δ), storing just the minimum and maximum latitudes and longitudes essentially stores the positions of all reference latitudes and longitudes and fulfils all our algorithmic requirements. The minimum (and maximum) latitude (and longitude) can be identified by a single scan of the batch of position fixes (i.e. latitude, longitude tuple) used for generating the map.

2) Matrices for storing density contributions: Remember from the discussion in III-C1 that we calculate the density contributions only at the centeres of the tiles. For storing the density contributions at the centers, we use a 2-D rxc matrix, $\mathcal{V}$. Here r and c are:

$$r = \lceil \frac{max\_lat - min\_lat}{\Delta} \rceil \tag{9}$$

$$c = \lceil \frac{max\_long - min\_long}{\Delta} \rceil \tag{10}$$

We call the matrix $\mathcal{V}$ as *vacancy matrix* and each of it's element, ν_{ij}, stores the density contribution at the center, c_{ij}, of the tile, T_{ij}. We use another rxc matrix $\mathcal{N}$, called the *normalization matrix*, which stores the normalization coefficients corresponding to the stored density evaluations. We initialize all elements of $\mathcal{V}$ with 0 and all elements of $\mathcal{N}$ with 1.

3) Representing density contribution of a position fix: Recall from section III-C2 that density contribution of each position fix is calculated using it's corresponding f_i. Given latitude (lat_i), longitude ($long_i$) and co-variance matrix of uncertainty (Σ_i), the corresponding f_i is given by

$$f_i(lat, long) = \frac{1}{\sqrt{(2\pi)^2 |\Sigma_i|}} e^{-\frac{1}{2} L_i^T \Sigma_i^{-1} L_i} \tag{11}$$

where

$$L_i(lat, long) = \begin{pmatrix} lat \\ long \end{pmatrix} - \begin{pmatrix} lat_i \\ long_i \end{pmatrix} \tag{12}$$

Fig. 2. Skewed grid formed using latitudes and longitudes used in our experiment to map the hallway of Bharati building (IIT Delhi). Visualized using Google Earth.

We are interested in finding the density contributions only at discrete points, i.e. the centers of the map tiles. Hence we represent the continuous f_i as a matrix, $\mathcal{K}$. Since $\mathcal{K}$ is the matrix representation of a 2-D Gaussian, it contains the symmetric structure of Gaussian distribution (see Fig. 3 for sample) and hence is indexed about the center-most cell. The center-most cell is indexed (0,0), the cell just right to the center-most cell is indexed as (0,1), cell just left to the center-most is indexed (0,-1), cell vertically at the top of center-most is indexed (1,0) and so on. Following this indexing pattern, the value of the element k_{ij} of matrix $\mathcal{K}$ is given by:

$$k_{ij} = \frac{1}{\sqrt{(2\pi)^2|\Sigma_i|}} e^{-\frac{1}{2}\begin{pmatrix} i.\Delta \\ j.\Delta \end{pmatrix}^T \Sigma_i^{-1} \begin{pmatrix} i.\Delta \\ j.\Delta \end{pmatrix}} \tag{13}$$

f_i extends to infinity but it's matrix representation, $\mathcal{K}$, has fixed dimensions. To limit the dimensions of the matrix, an error tolerance parameter ϵ is chosen. The density contribution by f_i at a point (lat,$long$) is effectively ignored if it is below ϵ. The dimensions of $\mathcal{K}$ can now be chosen as $(2r_k+1)$ x $(2c_k+1)$ such that $k_{(r_k+1)0}$, $k_{(-r_k-1)0}$, $k_{0(c_k+1)}$ and $k_{0(-c_k-1)}$ are all just below ϵ. Therefore, solving inequality with ϵ using equation (13), r_k and c_k are given by:

$$r_k = \lfloor \sqrt{\frac{-2}{\Delta^2.\Sigma_{i_{22}}} log(\sqrt{4\pi^2|\Sigma_i|}.\epsilon)} \rfloor \tag{14}$$

and

$$c_k = \lfloor \sqrt{\frac{-2}{\Delta^2.\Sigma_{i_{11}}} log(\sqrt{4\pi^2|\Sigma_i|}.\epsilon)} \rfloor. \tag{15}$$

$$\begin{pmatrix} 0.0033 & 0.0478 & 0.1163 & 0.0478 & 0.0033 \\ 0.0081 & 0.1163 & 0.2829 & 0.1163 & 0.0081 \\ 0.0033 & 0.0478 & 0.1163 & 0.0478 & 0.0033 \end{pmatrix}$$

Fig. 3. An example of matrix $\mathcal{K}$

4) Accumulating density contributions: The final *pdf* of the map or the probabilistic map is generated by accumulating the density contributions by all position fixes at the canters of the tiles, c_{ij}. We already know that the density contribution of each position fix is represented by the corresponding matrix $\mathcal{K}$ and the contributions are stored in the vacancy matrix, $\mathcal{V}$. The process of accumulating density contributions is iterative : One position fix (along with its associated uncertainty) is chosen at a time, it's $\mathcal{K}$ matrix is formed which is then used to update the *vacancy matrix*, $\mathcal{V}$. At each iteration, the elements of $\mathcal{V}$ are updated as follows (see Figs. 4 and 5). The matrix $\mathcal{K}$ is imagined to be placed over the matrix $\mathcal{V}$ such that the center of the smaller matrix $\mathcal{K}$ (cell with index (0,0)) is positioned over the cell (p, q) of $\mathcal{V}$. Here p, q is the subscript of the map tile T_{pq} within the bounds of which the position fix represented by $\mathcal{K}$ (i.e lat_i, $long_i$) lies. These are given by

$$p = \lfloor \frac{lat_i - min\_lat}{\Delta} \rfloor \tag{16}$$

$$q = \lfloor \frac{long_i - min\_long}{\Delta} \rfloor. \tag{17}$$

Once this alignment of both matrices have been done, the respective overlapping elements are added to one another and stored back into the cells of the matrix $\mathcal{N}$.

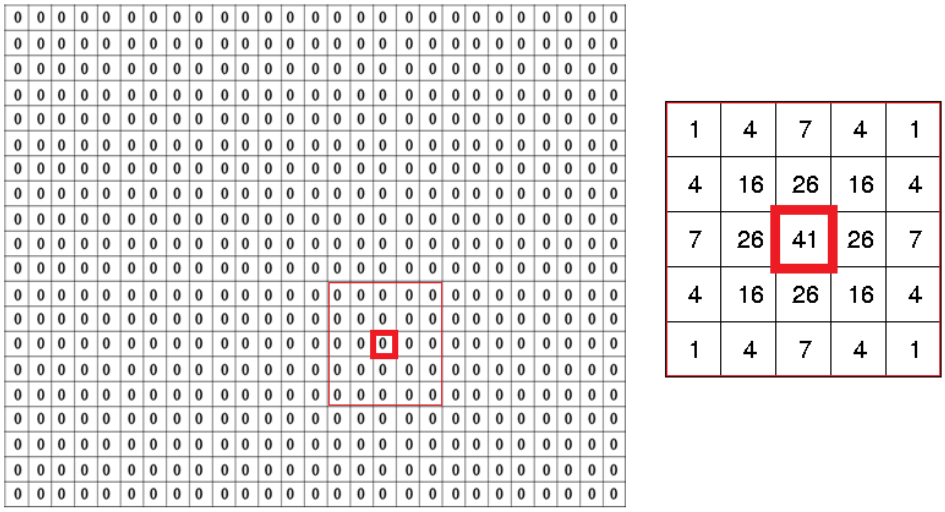


Fig. 4. The smaller matrix to the right, represents the $\mathcal{K}$ matrix for a position fix. The larger matrix to the left represents vacancy matrix $\mathcal{V}$. The red boundaries show the imaginary placement of matrix $\mathcal{K}$ over $\mathcal{V}$. The center of $\mathcal{K}$, marked with broad red borders, is placed over broad red-bordered cell of $\mathcal{V}$. The updated vacancy matrix is shown in Fig. 5. The values shown here are imaginary.

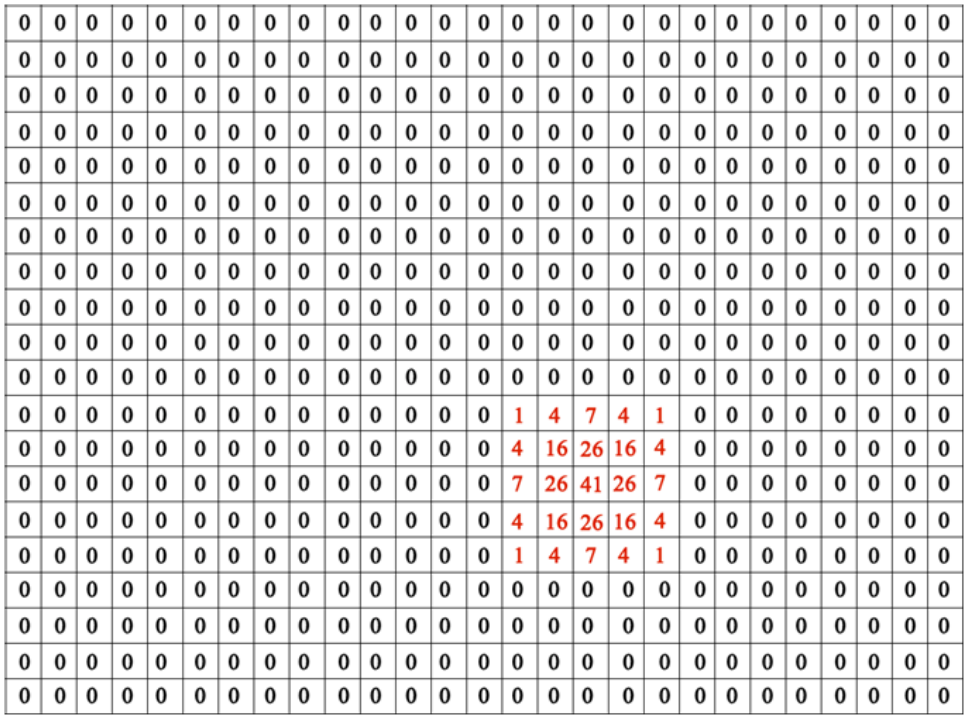


Fig. 5. Updated vacany matrix, $\mathcal{V}$ at the end of 1^{st} iteration. The values shown here are imaginary.

At every iterative step, for every cell of matrix $\mathcal{V}$ that is updated the corresponding cell of the normalization matrix $\mathcal{N}$ is incremented by 1. At the end of all iterations, all the elements of the vacancy matrix are divided by the elements of the normalization matrix in an element-by-element manner.

$$\nu_{ij\ final} = \nu_{ij}/n_{ij} \tag{18}$$

The final vacancy matrix stores the desired probabilistic map.

B. Generating building floorplan

We use the probabilistic map, generated in the first stage, to demarcate the position of walls and other obstacles. In the probabilistic version, every tile is associated with the probability of it being vacant. But for the building floorplan, we want to get rid of the probabilities and clearly classify each tile as being either vacant or occupied by wall (or any other obstacle). Since the final *vacancy matrix* already stores

Algorithm 1 Generating probabilistic map

Input: $\mathcal{P}, \Delta$
Output: $\mathcal{V}, min\_lat, min\_long$
  $min\_lat \leftarrow$ {min latitude in $\mathcal{P}$} - tolerance
  $min\_long \leftarrow$ {min longitude in $\mathcal{P}$} - tolerance
  $max\_lat \leftarrow$ {max latitude in $\mathcal{P}$} + tolerance
  $max\_long \leftarrow$ {max longitude in $\mathcal{P}$} + tolerance
  Initialize $\mathcal{V}$ matrix with all 0's
  Initialize $\mathcal{N}$ matrix with all 1's
  for all (lat,long,σ) tuples in $\mathcal{P}$ **do**
    Form $\mathcal{K}$ matrix
    Update $\mathcal{V}$ matrix
    Update $\mathcal{N}$ matrix
  end for
  $\mathcal{V} \leftarrow \mathcal{V}./\mathcal{N}$

values in the normalized form, the classification is easily done by applying a common global threshold. The tiles whose corresponding values in the vacancy matrix are below the chosen threshold are marked as occupied and the rest as vacant. Rendering the final map with occupied locations (or tiles) marked differently from the pedestrian usable locations, is surely an acceptable way of map representation. But within the limits of our experimentation method we have found that, when the map is depicted in the above manner, the implied boundaries of the hallways appear highly irregular and noisy. The discretization of the map into tiles may be a primary reason for this. Therefore, we go a step further and determine and then regularize the boundaries of the obstacles. We render the final map as walkable sections enclosed and demarcated by boundaries to obstacles.

For marking the boundaries, we first assimilate together the corner points of the *vacant* tiles and then find the α-shape [9] of the 2-D point cloud. α-shapes are essentially composite shapes formed by a chain of piecewise linear strokes, where the parameter α controls the length of each stroke. They are a generalization of the convex-hull and can be used to determine the shape of any finite set of points. Lowering the value of the parameter α produces more regularized boundaries. But this regularization, which may be needed in one part of the map, leads to the loss of intricate structure in other parts of the map. So, we propose to use an intermediate value of α and then apply additional heuristics to regularize all sections of map boundaries without the loss of structural intricacy. At this point, we cannot argue about the best value of α, but based on our experimentation, we recommend choosing $\frac{1}{\alpha}$ as 3-5 times the width of the map tiles (i.e Δ) and then applying the heuristic defined below.

We define for ourselves a small angular tolerance, τ (some value between 5-15 degrees is recommended). We start with one of the linear stokes of the composite α-shape. We keep on accepting subsequent strokes in the chain as long as their angles, relative to the initially chosen stroke, lie within $\pm\tau$. When a subsequent stroke violates the angular tolerance, we stall the process of accepting strokes, replace all currently accepted strokes with a single linear stroke joining the endpoints and then again repeat the process starting with the previously unaccepted stroke (i.e. the stroke that stalled the process). This way, we replace sets of smaller strokes with larger strokes thus regularizing the boundaries, but at the same time preserve the intricate structures wherever required.

V. Experiments

We tested our mapping algorithm by estimating the shape of the hallway of *Bharati* building in IIT Delhi campus. As mentioned earlier, the algorithm is independent of the positioning device or method used. But for contextual purposes, we first present a qualitative overview of the positioning method used before discussing the procedure and results in detail.

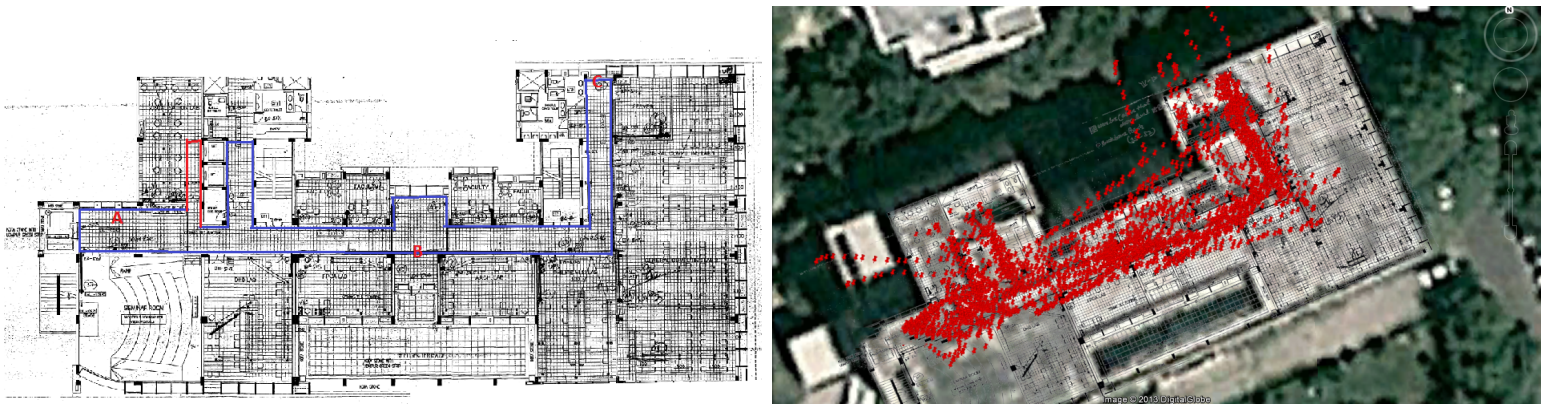
A. Overview of positioning method

We implemented a *dead reckoning* [10] positioning algorithm which estimates the final position of a pedestrian, given his initial position, by calculating the net displacement. The entire algorithm can be divided into 3 broad parts: heading estimation, step detection and stride length estimation.

1) Heading estimation: For heading estimation we use the work on robust attitude estimation of hand-held positioning systems by Mahesh Chowdhary *et al.* [11], making implementation level changes wherever necessary. The complete attitude of the smartphone is estimated by fusing the readings of gyroscope as well as magnetometer. The fusion is carried out by an Extended Kalman Filter (EKF) which uses an S-shaped curve to adaptively change the measurement noise matrix (R) and the system noise matrix (Q). The heading estimated by this method is claimed to be robust and resilient to arbitrary human actions like swinging of hands while walking.

2) Step detection and stride length estimation: The exact algorithms used for step detection and stride length estimation are proprietary of CSR Technologies Pvt. Ltd. Both these algorithms make use of the absolute accelerations measured with respect to earth-centered, earth-fixed (ECEF) frame of reference. But, the accelerations measured by the 3-axis accelerometer in a smartphone are relative to the phone's orientation. Estimating the full 3-D attitude of the smartphone during heading estimation (refer to section V-A1), enables us to transform the measured accelerations relative to the smartphone's frame of reference to the required absolute accelerations in ECEF frame. This transformation is brought about by the multiplication of the 3-D vector of relative accelerations with a rotational transformation matrix whose elements can be calculated using the heading, pitch and roll of smartphone's orientation.

We do not numerically characterize the performance of the positioning method implemented and rather acknowledge that more accurate and efficient techniques do exist. But the ability to use readings from such sub-optimal positioning methods, to generate maps as accurately as possible, is a desirable characteristic of any crowd-sourcing mapping algorithm. As results indicate, our mapping algorithm does have this important characteristic.



(a) Reference floorplan

(b) 100 representative trajectories

Fig. 6. (a) Reference floorplan of the Bharati building. The blue and red lines mark the present true boundaries of the hallway. A, B and C are the starting points of all trajectories in our dataset. (b) 100 representative trajectories are visualized using Google Earth. The red dots are individual position fixes.

B. Experimental procedure

We implemented the positioning algorithm, described above, in an off-the-shelf Samsung Galaxy S-III smartphone as an android application. The android application, curtailed for our experiment, records pedestrian trajectories as log files stored in the smartphone. These log files are later downloaded into a computer and our mapping algorithm is run to generate the map. In our experiment, the starting point of each trajectory was calculated using GPS fix. Therefore, we chose only those 3 positions as starting points where GPS fix was readily available, for example, entrance to the adjoint terrace, hallway corners with large windows (see letter labels in figure 6a).

We collected 300 trajectories of varied lengths and shapes to generate the final map presented. Equal number of trajectories (100 each) were recorded with the smartphone held steadily in hand, smartphone placed in trouser pocket and smartphone in hand with arms swinging. A representative set of 100 trajectories has been shown is figure 6b. As it is evident from the visualization, some of the trajectories have arbitrarily large errors. The primary reason for these large errors, we believe, is significant heading error induced by the momentary loss of magnetometer caliberation in the presence of strong magnetic disturbances at certain points of the hallway. Apart from these visibly erroneous trajectories, we have also found that a majority of the trajectories recorded, have significant deviation from the original path traversed. But such errors can be expected from any PDR positioning algorithm without the assistance of error-resetting techniques.

We evaluate the generated probabilitic map qualitatively by comparing it to the original floorplan of the building (figure 6a). The reference boundary to the hallway (against which the algorithmically estimated boundary is compared in figures 8 and 9) is drawn to scale by precisely overlaying the floorplan over *Bharati* building using Google Earth. The reference floorplan used, was prepared when the building was initially constructed and does not show an extension of the hallway that was later done (the section bounded by red boundaries in figure 6a). This extension of hallway, as we show in figures 8 and 9, is captured by our algorithmically generated map thus highlighting an important utility of automatically maintaining indoor maps: keeping floorplans up-to-date.

C. Results

The skewed grid used for representating the map in our experiment is visualized in figure 2. The minimum and maximum latitudes in the grid are 28.544733 degrees North and 28.545303 degrees North respectively, the minimum and maximum longitudes in the grid are 77.189950 degrees East and 77.190811 degrees East respectively. The interval between adjacent latitudes (or longitudes), i.e. Δ, is chosen to be $6\text{x}10^{-6}$ degrees. The map tiles , therefore, have a size of about 66 cm x 66 cm.

The vacancy matrix ($\mathcal{V}$) and the normalization matrix ($\mathcal{N}$) have dimensions 95 x 144, calculated using equations (9) and (10). The implemented positioning android application logs

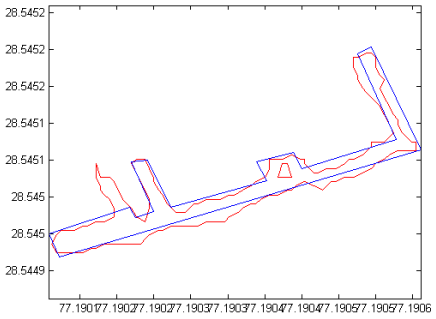
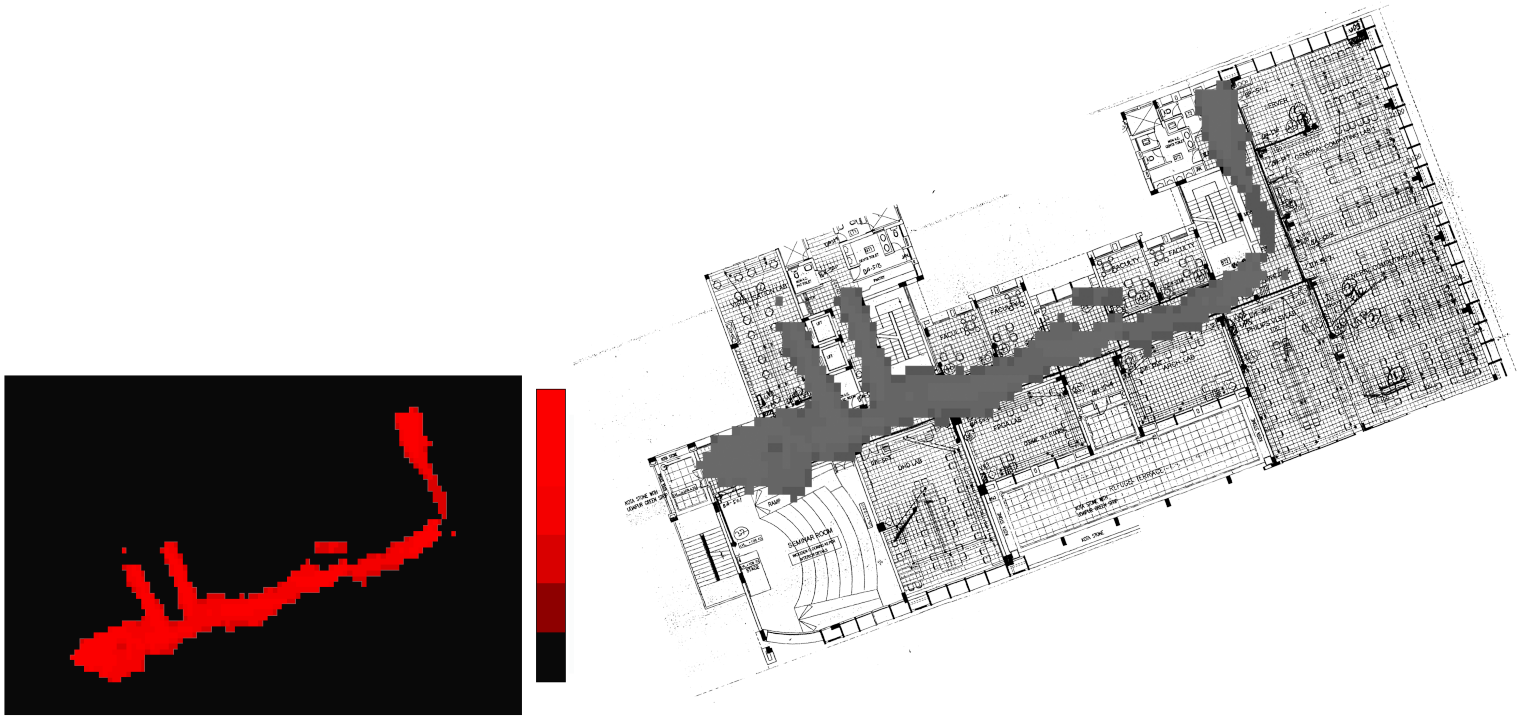


Fig. 8. Estimated boundaries of the hallway obtained using α-shapes. The red lines represent the estimated boundaries, the blue lines represent the boundaries depicted in the reference floorplan. Notice that the outdated reference floorplan does not show an extension in the hallway which the estimated boundaries accurately incorporate. The small loop in the center of the hallway represents the estimated boundaries of small sitting arrangement located there but not incorporated in the reference floorplan. The boundaries are drawn to scale.



(a) Thresholded vacancy matrix

(b) Predicted vacant map tiles

Fig. 7. (a) The thresholded vacany matrix. The various colours in the heatmap are arranged in the colourbar, from bottom to top, in an increasing order of the sets of values that they represent. (b) The prediced vacant tiles are shaded in dark colours and overlaid over the reference floorplan.

uncertainties as tuples : (deviation in latitude, deviation in longitude). We convert these tuples to the required 2x2 covariance matrix format in the manner described in system overview (refer to section II).

The final vacancy matrix ($\mathcal{V}$) obtained, is used to classify the vacant map tiles from the occupied ones. The classification is done by choosing a common global threshold which is 0.4 in our case. The thresholded vacancy matrix, obtained by replacing the values below the threshold with 0 and leaving the the values above the threshold unchanged, is shown in figure 7a as a contour plot (or heatmap). The predicted vacant map tiles overlaid over the reference floorplan, is presented in figure 7b for qualitative comparison. Having identified the vacant tiles, we estimate the boundaries using α-shapes with $\frac{1}{\alpha}$ set as 200 cm. We use Delaunay traingulation for the computation of the α-shapes. The estimated boundaries using α-shapes is presented in figure 8. The boundaries formed are highly irregular and hence we apply our boundary regularization heuristic (discussed in the last paragraph of the section IV-B). The final estimated map with regularized boundaries is shown in figure 9.

VI. Related Work

In this section, we review three prior works which we believe are closely related to our contribution and, in the process, highlight the importance of our work.

SmartSLAM [12] is a system which extends the Simultaneous Locatization and Mapping (SLAM) technique to smartphones. SLAM is a technique used to generate the map of an unknown location while localizing the pedestrian at the same instant. SmartSLAM uses inertial sensors as well as WiFi signals to construct RF fingerprints and building floorplans. But, like any SLAM technique, SmartSLAM requires the pedestrian to walk in loops i.e. revisit previously visited places, so that convergence of the map estimate is possible. This might be an onerous responsibility for the general pedestrians. Also, since

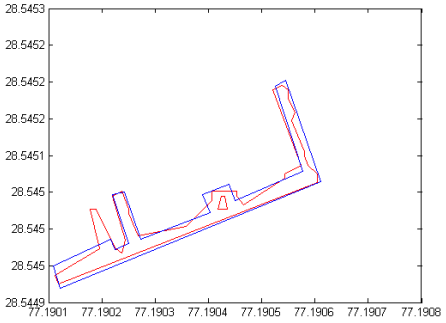


Fig. 9. Regularized boundaries using the proposed boundary-regularization heuristic. Red lines mark the estimated boundaries, blue lines mark the boundaries drawn to scale from the reference floorplan.

localization and mapping are inter-dependent in SmartSLAM (any any SLAM algorithm), using it for crowdsourcing indoor maps inherently enforces the restriction of using the same navigation application by all participating pedestrians. Such a restriction cannot be practically ensured at a large scale, thus reducing SmartSLAM's viability as crowdsourcing method for automatic maintenance of map database.

CrowdInside [13] is a crowdsourcing-based system for automatically maintaining indoor floorplans. The mapping algorithm segments the motion traces of pedestrians and then uses α-shapes to determine the shape of hallway and rooms. This mapping technique is highly dependent on the accuracy of the motion traces. It fails in the presence of highly erroneous motion traces like those in our experiment dataset (refer to figure 6b). To ensure highly accurate motion traces, it uses a PDR positioning algorithm which utilizes landmark anchors (like elevators, escalators etc). Firstly, such landmark anchors may not be available in many places. Secondly, by limiting the positioning algorithm compatible with the mapping technique, the participation of general pedestrians in crowdsourcing is restricted to only those who are using that particular positioning algorithm.

Yifei Jiang *et al.* proposed a system for automatic floorplan construction [14], which overcomes most of the challenges of the above systems. The system uses WiFi fingerprints to identify and enumerate rooms and inertial sensor readings to estimate the shape of hallways. However, the system has its own shortcomings. It works only for rectangular shaped rooms. It does not include the positions of furniture into the floorplan. Finally, it is highly dependent on the availability of WiFi fingerprints.

Keeping in mind the 3 different prior works discussed above, we present the major contributions of our proposed mapping algorithm. Our algorithm removes the requirement of conformity among the positioning methods used to collect data. It can work with any positioning method or device as long as the required data is supplied. It is robust to the varying amounts of error that might be present in the dataset and always generates the best possible map given the data. It does not expect loop-closure or other special forms of participation from the users. It only requires pedestrians to walk into the building with positioning system switched on (to get the initial position fix from GPS data) and navigation applications to keep sharing the positioning data they calculate. The algorithm does not depend on the availability of WiFi signals and gives a more complete picture of the building floorplan by incorporating the position of walls, furniture or other obstacles to pedestrian motion in the floorplan.

VII. Conclusion

In this paper, we present a crowdsourcing based server-side mapping algorithm which can accept data from any positioning device and method. The algorithm can be used to automatically maintain a database of indoor floorplans. We present a novel way to approach the mapping problem using *spatial density histograms with varied kernels* which, by virtue of its origin from kernel density estimation, promises the convergence of the estimated map with the actual map as number of input samples tend to ∞. The underlying concept also ensures that the mapping algorithm is robust to the varying amounts of error present in the input positioning data. The results from our experiment, conducted in a building of our university, qualitatively validate the robustness of the mapping algorithm to arbitrarily high errors in positioning data supplied.

Acknowledgment

This work was supported by project RP02732 "Tracking Human Motion via Sensor Fusion" funded by CSR Technologies India Pvt. Ltd.

References

[1] S. Wan and E. Foxlin, "Improved pedestrian navigation based on drift-reduced mems imu chip," in *Proceedings of the 2010 International Technical Meeting of The Institute of Navigation*, 2001, pp. 220–229.

[2] M. Angermann and P. Robertson, "Footslam: Pedestrian simultaneous localization and mapping without exteroceptive sensorshitchhiking on human perception and cognition," *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1840–1848, 2012.

[3] R. Van Renesse, K. P. Birman, and S. Maffeis, "Horus: A flexible group communication system," *Communications of the ACM*, vol. 39, no. 4, pp. 76–83, 1996.

[4] M. Alzantot and M. Youssef, "Uptime: Ubiquitous pedestrian tracking using mobile phones," in *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*. IEEE, 2012, pp. 3204–3209.

[5] "Skyhook Wireless Inc." www.skyhookwireless.com.

[6] M. Azizyan, I. Constandache, and R. Roy Choudhury, "Surroundsense: mobile phone localization via ambience fingerprinting," in *Proceedings of the 15th annual international conference on Mobile computing and networking*. ACM, 2009, pp. 261–272.

[7] B. W. Silverman, *Density estimation for statistics and data analysis*. CRC press, 1986, vol. 26.

[8] L. Devroye and L. Gyorfi, *Nonparametric Density Estimation: The L_1 View*. Wiley, 1985.

[9] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," *Information Theory, IEEE Transactions on*, vol. 29, no. 4, pp. 551–559, 1983.

[10] L. Ojeda and J. Borenstein, "Personal dead-reckoning system for gps-denied environments," in *Safety, Security and Rescue Robotics, 2007. SSRR 2007. IEEE International Workshop on*. IEEE, 2007, pp. 1–6.

[11] M. Chowdhary, M. Sharma, A. Kumar, S. Dayal, and M. Kumar, "Robust attitude estimation for indoor pedestrian navigation application using mems sensors," in *Proceedings of the 25th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2012)*, 2001, pp. 1658–1665.

[12] H. Shin, Y. Chon, and H. Cha, "Unsupervised construction of an indoor floor plan using a smartphone," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 42, no. 6, pp. 889–898, 2012.

[13] M. Alzantot and M. Youssef, "Crowdinside: automatic construction of indoor floorplans," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. ACM, 2012, pp. 99–108.

[14] Y. Jiang, Y. Xiang, X. Pan, K. Li, Q. Lv, R. P. Dick, L. Shang, and M. Hannigan, "Hallway based automatic indoor floorplan construction using room fingerprints," in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. ACM, 2013, pp. 315–324.