Cooperative Localization with Hybrid INS/PDR Tracking for GPS-denied Environments*

Author 1 and Author 2
Address 1
Affiliation 1
Country 1
email 1

Author 3
Address 2
Affiliation 2
Country 2
email 2

Author 3
Address 3
Affiliation 3
Country 3
email 3

ABSTRACT

In this paper, we develop a novel probabilistic hybrid inertial navigation system (INS)/pedestrian dead reckoning (PDR) measurement tracking algorithm, namely (PHIMTA), that provides high accuracy tracking in slow pedestrian scenarios. We then combine it with the latest localization algorithms, such as grid-based belief propagation (GBP) and stop-and-go (SnG), that allow for improved accuracy in GPS-denied environments.

Keywords

ACM proceedings; Cooperative localization; Mobility; INS

CCS Concepts

•**Mathematics of computing** → **Bayesian computation;** •**Human-centered computing** → **Mobile devices;** •**Computing methodologies** → *Mixture models;*

1. INTRODUCTION

Coping with mobility has been key in localization research. The typical scenario is to complement GPS with information from inertial measurement units (IMUs) and odometers to provide uninterrupted navigation solutions during GPS outages. The integration is typically achieved by a Kalman filter (KF) [1], or some variants, e.g., the extended Kalman filter (KF) [2]. Unfortunately, the significant errors of micromechanical systems (MEMS) inertial sensors as well as the time-varying models cannot be modelled accurately by the KF linearized models. As an alternative to better capture the non-linearities, the use of a particle filter has been proposed [3].

The information from the sensors provides all the necessary information required for tracking of human movement.

*This work is supported by XXX.

SAC'16, April 4-8, 2016, Pisa, Italy

http://dx.doi.org/xx.xxxx/xxxxxxx.xxxxxxx

There has been much research on pedestrian dead reckoning (PDR) and its applications. In PDR, the frequency of the pedestrians steps is extracted from the sensor information and assuming some statistical model for the length and course of the steps, the pedestrians direction and distance travelled are calculated by summing up all the steps, e.g., [4]. There has also been extensive research in the classification and modelling of sensor outputs with different human movement, e.g., see [5,6], but the reality is that erratic movement, e.g., walking in slopes, or abrupt movements, that can typically occur in a battlefield, will make the error grow quickly out of hand.

Alternatively, pedestrian tracking can be treated as an application of a strap-down inertial navigation system (INS). In this case, the orientation of a sensor module is tracked by integrating the angular velocities, which are subsequently used to determine the acceleration components in the GCS. Then the gravity acceleration is subtracted and the remaining acceleration is integrated over time to find the sensors displacement. Unfortunately, low cost MEMS-IMUs are susceptible to errors, such as misalignment errors, scale factor, bias turn-on error, bias drift error, and etc. Though deterministic errors can typically be removed via calibration, stochastic errors cannot be removed and can increase quickly. Analysis and modelling of the MEMS-IMU errors can be found in [3, 7, 8].

A solution proposed in [9] has been to provide a synergism between PDR and INS. Essentially the movement of the pedestrian will be calculated by finding the orientation and number of steps as in PDR, but the characteristics will be derived from the IMU measurements instead of using a statistical model.

The combination of PDR localization and cooperative localization for GPS denied environments however has not been well investigated. The SPAWN framework in [10] considered mobility, but it was demonstrated in [11] that it is too computationally expensive for real-world hand-helds and a heuristic cooperative localization algorithm, called stop-and-go (SnG), was proposed as an alternative. SnG keeps the computational cost low for mobile devices while synergizing with PDR. Still due to the heuristics in SnG, the network is highly susceptible to node placement and if the placement is not uniform enough, then the whole network localization will collapse.

This paper's aim is to develop a robust low-cost algorithm

for pedestrian mobility tracking in GPS-denied environments. We first devise a dynamic Bayesian network in which we generalize the INS/PDR pedestrian tracking algorithm in [9] by using probabilistic particle representations and then combine it with a recent cooperative localization algorithm, namely grid-based belief propagation (GBP) [12], to have a fully distributed and robust probabilistic model for mobile pedestrian tracking which combines low computational cost, as well as robustness in different node geometries, and high localization accuracy.

Our contributions can be summarized as follows:

- We design a probabilistic pedestrian tracking technique which is referred to as the particle hybrid inertial measurement tracking algorithm (PHIMTA).
- Also, we combine PHIMTA with GBP, creating a novel dynamic Bayesian network model most suitable for mobile localization in GPS-denied environments.[1]
- We conduct Monte-Carlo simulations using the real data in [13, 14] that show that PHIMTA/GBP provides consistently equivalent accuracy with drastically decreased computational cost, compared to the literature.

2. PROBLEM FORMULATION

We consider a network of nodes in a 2D environment which consists of N agents and M anchors, where $M \geq 4$ and $N \gg M$. Let $\boldsymbol{X} = [X_1, \ldots, X_i, \ldots, X_{N+M}]$ be the locations of all nodes, with X_i representing the unique identifier of node i and $X_i \in \{x_1, x_2, \ldots x_k\}$, where k iterates over all possible IDs. Also, let $\boldsymbol{Z}$ denote the coordinates of all nodes, with $\boldsymbol{Z}_i$ representing the coordinates of node i, and the domain of $\boldsymbol{Z}_i$ is $\Re^2$. The nodes communicate wirelessly and it is assumed that the maximum communication range for each node is $R_{\max}$. Time is slotted and time slots are denoted by the time index superscript (t) for $t = 1, 2, \ldots, \infty$.

Based on the behaviour of the nodes, there are two types of time slots. Firstly, the nodes might move and use IMU information to update their information, namely IMU time slots, or they might stay idle and cooperate with their neighbours to update their location estimate, namely belief propagation (BP) time slots. The nodes use cooperative localization every n time slots, while in between they have a probability $p(W^{(t)})$ at each time slot to wait or to move. If a node is moving during a BP time slot, then it will not participate in the message passing algorithm. It uses the SHOE filter, cf. Section 4, to discriminate between being idle or not.

Let $p(X_i^{(t)})$ be the probability density function (pdf), i.e., the belief that node i has about its location at time t. We model $p(X_i^{(t)})$ as a multinomial distribution with parameters θ_k, where θ_k is the probability of node i being in ID x_k and $\sum_k \theta_k = 1$. Let $p(\boldsymbol{X}^{(t)})$ denote the state of the system at time slot (t). We assume that the system is Markovian and represent it as a pdf. Then we have

$$P(\boldsymbol{X}^{(0)}, \boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(t)}) = p(\boldsymbol{X}^{(0)}) \prod_{\tau=0}^{t-1} p(\boldsymbol{X}^{(\tau+1)} | \boldsymbol{X}^{(\tau)}), \tag{1}$$

where $p(\boldsymbol{X}^{(0)})$ is the initial system state and $p(\boldsymbol{X}^{(\tau+1)}|\boldsymbol{X}^{(\tau)})$ is the transition probability. Depending on the type of time slot the transition probability will change. Thus, we have

$$p(\boldsymbol{X}^{(\tau+1)}|\boldsymbol{X}^{(\tau)}) = \begin{cases} p(\boldsymbol{X}^{(\tau+1)}|\boldsymbol{X}^{(\tau)}, \boldsymbol{R} = \boldsymbol{r}), \text{for BP time slots,} \\ p(\boldsymbol{X}^{(\tau+1)}|\boldsymbol{X}^{(\tau)}, \boldsymbol{O} = \boldsymbol{o}), \text{for IMU time slots,} \end{cases} \tag{2}$$

in which $\boldsymbol{r}$ denotes a vector with all the distance measurements between nodes, and $\boldsymbol{o}$ is a vector with the IMU observations. The above can be described graphically in Fig. 1.

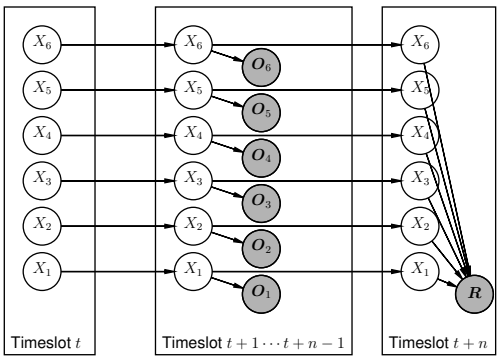

Figure 1: A dynamic Bayesian network model.

During the BP time slots, let the set of all nodes within the range of node i be the neighbourhood $\mathcal{N}_i$. Initially, the belief for the agents can be a non-informative uniform pdf over the grid, while the anchors' pdfs are focused in the IDs close to the real position, e.g., within 10m. Node i receiving a message from node j at time slot t can derive, using time-of-arrival (ToA) measurements,[2] a noisy estimate $r_{j\to i}^{(t)}$ of the distance between them. For convenience, we assume $r_{j\to i}^{(t)} = r_{i\to j}^{(t)} = r_{ji}^{(t)}$.

Thus, as in [15], for ToA distance measurements, we define the random variable $R_{ji}^{(t)}$ with its value $r_{ji}^{(t)}$ modelled as

$$r_{ji}^{(t)} = \|\boldsymbol{z}_i^{(t)} - \boldsymbol{z}_j^{(t)}\| + \eta_{ji}, \tag{3}$$

where η_{ji} is a Gaussian noise with variance $\sigma_{ji}^2 = K_e \|\boldsymbol{z}_i^{(t)} - \boldsymbol{z}_j^{(t)}\|^{\beta_{ji}}$ in which K_e is a proportionality constant capturing the combined physical layer and receiver effect, and β_{ji} denotes the path loss exponent. In the case of line-of-sight (LoS), η_{ji} is assumed zero mean, and $\beta_{ji} = 2$, i.e., $\eta_{ji} \sim \mathcal{N}(0, \sigma_{ji}^2)$.

[1] We choose Grid-BP due to the low computational cost, but any message passing variant can be used.

[2] The assumption of using ToA is not restrictive on the proposed algorithm because it can easily be used with other measurement models.

We define the likelihood of node i and node j measuring distance $R_{ji}^{(t)} = r_{ji}^{(t)}$ between them at time t, given X_i, X_j as

$$p(R_{ji}^{(t)} = r_{ji}^{(t)} | X_i^{(t)}, X_j^{(t)}), \propto \exp\left(-\left(\frac{r_{ji}^{(t)} - \|C_i^{(t)} - C_j^{(t)}\|_2}{h}\right)^2\right), \quad (4)$$

where h controls steepness, $C_i^{(t)}$ and $C_j^{(t)}$ are the coordinates of the centres of the grids' squares $X_i^{(t)}$ and $X_j^{(t)}$, respectively. Thus, we aim to find the maximum a posteriori (MAP), i.e., the values that maximize $p(\boldsymbol{X}^{(t+1)}|\boldsymbol{R}^{(t+1)}, \boldsymbol{X}^{(t)})$ given distance measurements $\boldsymbol{R}^{(t+1)} = [R_{ji}^{(t+1)}]$. For node i, we have

$$\hat{X}_i = \arg\max_{X_i} p(X_i^{(t+1)} | \boldsymbol{R}_i^{(t+1)}, X^{(t)}). \quad (5)$$

Consequently, $p(X_i^{(t+1)}|\boldsymbol{R}_i^{(t+1)}, X_i^{(t)})$ can be evaluated using the Bayes' rule as

$$\begin{aligned} &p(X_i^{(t+1)}|\boldsymbol{R}_i^{(t+1)}, X_i^{(t)}) \\ &\propto p(X_i^{(t)}) \prod_{j\in\mathcal{N}_i} p(R_{ji}^{(t+1)}|X_i^{(t+1)}) \\ &\propto p(X_i^{(t)}) \prod_{j\in\mathcal{N}_i} \int p(R_{ji}^{(t+1)}|X_i^{(t+1)}, X_j^{(t+1)}) p(X_j^{(t+1)}) \mathrm{d}X_j, \end{aligned} \quad (6)$$

in which the sign "$\propto$" means "is proportional to", and normalization should be done to obtain the pdf.

Similarly, during the IMU time slots, we have

$$p(X_i^{(t+1)}|\boldsymbol{O}_i^{(t+1)}, X_i^{(t)}) \propto p(X_i^{(t)}) p(\boldsymbol{O}_i^{(t+1)}|X_i^{(t+1)}). \quad (7)$$

The model is summarized in Algorithm 1.

Algorithm 1 Dynamic Bayesian Network

```
for all i ∈ N do
    Initialize p(X_i^(0)), v_i^(0), ω_i^(0), a_i^(0), μ_i^(0)
    for all t ∈ Time slots do
        if time slot is BP then
            calculate (2) using GBP [12, Algorithm 1]
        else
            calculate (2) using PHIMTA, Algorithm 2
        end if
    end for
end for
```

In the next sections, we will describe how GBP is used to solve (6) and then how PHIMTA solves (7).

3. BP TIME SLOTS

During the BP time slots, the network can be modelled as a cluster graph and we adopt a Bethe cluster graph [16]. The lower factors are composed of univariate potentials $\psi(X_i)$, while the upper region is composed of factors $\psi(X_i, X_j, R_{ji})$, e.g., see Fig. 2.

The lower factors are set to the initial beliefs for the given time slot (t), and the upper factors are set to the corresponding conditional pdfs (cpdfs):

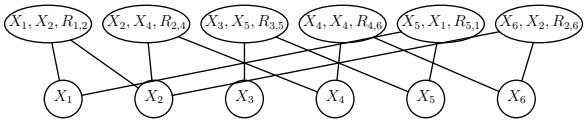

Figure 2: The cluster graph. Lower row factors denote the node position beliefs. Upper row factors denote the ranging interactions between the nodes.

$$\psi(X_i) = p(X_i^{(t)}), \quad (8)$$

$$\psi(X_i, X_j, R_{ji} = r_{ji}^{(t)}) = p^{(t)}(R_{ji} = r_{ji}^{(t)}|X_i, X_j). \quad (9)$$

Messages are then passed between nodes for multiple iterations until the node beliefs have converged. The message from node j to node i, at BP iteration $(s+1)$ is calculated by

$$\mu_{j\to i}^{(s+1)}(X_i) = \int \psi(X_i, X_j, R_{ji} = r_{ji}^{(t)}) \frac{b_{j\to i}^{(s)}(X_j)}{\mu_{i\to j}^{(s)}(X_j)} \mathrm{d}X_j, \quad (10)$$

where intuitively, a message (10) is the belief that node j has about the location of node i and $r_{ji}^{(t)}$ is the observed value of the distance between the nodes, at time slot t.

Then the belief of node i is updated as

$$b_i^{(s+1)}(X_i) = \lambda\psi(X_i) \prod_{k\in\mathcal{N}_i} \mu_{k\to i}^{(s+1)}(X_i) + (1-\lambda) b_i^{(s)}(X_i), \quad (11)$$

where λ is a dampening factor used to facilitate convergence.

BP continues until convergence, or if s reaches a maximum number of iterations $I_{\max}$. Then the beliefs, representing approximations to the true marginals, for each node are found by (11), i.e., $p(X_i^{(t+1)}) = b_i^{(s+1)}(X_i)$. Each node will need to perform a marginalization operation (10), and a product operation (11). Approximations are required for both complex operations. The details of the approximations, as well as more information on GBP can be found in [12].

4. IMU TIME SLOTS

During the IMU time slots, our is aim is to approximate the transition probability in (2). This is accomplished by using a non-parametric particle representation [16]. Assuming at time slot (t) the location pdf for node i is $p(X_i^{(t)})$, we represent it by a set of L random samples, or particles $\mathcal{S}_i^{(t)} = \{s_i^{(t,l)}\}_{l=1}^{L}$, sampled from $p(X_i^{(t)})$. The i-th sample is denoted as $s_i^{(t,l)} = (x_i^{(t,l)}, \pi_i^{(t,l)})$, where $x_i^{(t,l)}$ is the value of the node state and $\pi_i^{(t,l)} = \frac{1}{L}$ is the respective weight. Then the motion model is applied to each sample $s_i^{(t,l)}$ and we obtain a new sample

$$s_i^{(t+1,l)} = \left(x_i^{(t+1,l)}, \frac{1}{L}\right), \quad (12)$$

where

$$x_i^{(t+1,l)} \sim p(\boldsymbol{X}^{(t+1)}|X_i^{(t)} = x_i^{(t,l)}, \boldsymbol{O}_i^{(t)} = \boldsymbol{o}_i^{(t)}), \quad (13)$$

where $\boldsymbol{o}_i^{(t)}$ are the IMU measurements at time (t). Finally, the same parameter estimation [12, Algorithm 4] is used to obtain a multinomial parametric form of (2) from the samples.

4.1 PHIMTA

Our aim now is to derive an updated location particle given a location particle $x_i^{(t+1,l)}$ and the IMU observations $o_i^{(t)}$. We will derive the displacement, speed and attitude vectors, from the IMU sensors. We assume a typical MEMS sensor, consisting of an accelerometer, a magnetometer and a gyroscope. The measurements provided by the IMU and the magnetometer comprise the control input

$$\boldsymbol{o}^{(t)} = [\boldsymbol{a}^{(t)}, \boldsymbol{\omega}^{(t)}, \boldsymbol{\mu}_x^{(t)}],$$

and we denote their respective noise vector as

$$\boldsymbol{w}^{(t)} = [\boldsymbol{n}_a^{(t)}, \boldsymbol{n}_\omega^{(t)}, \boldsymbol{n}_\mu^{(t)}].$$

We assume that the input signal vectors from each sensor have length Y. Also, the noise is assumed to be white Gaussian noise and independent of previous states. A rotation matrix that maps the local coordinate system (LCS) of the sensors to global coordinates system (GCS) of the node is required, as the sensor axes may not match the nodes. Then the accelerometer observations will be mapped to the nodes coordinate system and used to calculate the displacement and speed of the node.

To alleviate the noise a number of schemes will be used. First is the fact that pedestrian walking is cyclical and significantly consistent. Each stride can be split into two phases. The stance phase, i.e., when the foot or part of the foot is placed on the ground, and the swing phase, i.e., when the foot is mid-air. Both the velocity and the angular velocity can be reset to zero at each stance phase, thus reducing the drift error accumulation. As the gyroscopes cannot be used in the static phase, signals from the accelerometer and magnetometer have to be used to calculate the orientations of the sensor module. To overcome tilt errors, the algorithm presented in [9] is used. The stance phase can be easily detected using peak detection, taking into consideration of the existence of zero crossings, e.g., [5]. In this work, we use the SHOE algorithm [17].

Hence, the system iterates through the following steps:

- Stance phase
  - Reset angular velocity to zero
  - Reset velocity to zero
  - Use magnetometer and accelerometer data to calculate rotation matrix
- Swing phase
  - Use gyroscope data to calculate the rotation matrix
  - Calculate the velocity and displacement using accelerometer data

Our derivation follows closely the work in [9]. In subsequent sections, as everything involves internal calculations at each agent, the node subscript is dropped for simplicity.

4.2 Coordinate Systems and Transformation Matrix

The global cartesian coordinate system used is the north-east-down (NED) frame (x^n, y^e, z^d). Consequently, the rotation matrix derived by using direction cosine representations is given by (14) (see top of next page), where p, r, a, correspond to the pitch, roll, and attitude, respectively, and the time slot superscript has been dropped for simplicity.

4.3 Swing Phase

During the swing phase, the orientation of a moving object is tracked by integrating the angular velocity vector $\boldsymbol{\omega}^{(t)} = [\omega_x^{(t)} - n_{\omega x}^{(t)}, \omega_y^{(t)} - n_{\omega y}^{(t)}, \omega_z^{(t)} - n_{\omega z}^{(t)}]$, obtained from the gyroscope after we correct for noise. Let the sampling period δt be short and $\delta\boldsymbol{\Psi} = [\delta a, \delta p, \delta r]$ be the rotated angle vector of the sensors. Then $\delta\boldsymbol{\Psi} = \boldsymbol{\omega}\delta t$. Assuming a small δt the rotation matrix for a period can be approximated by

$$\boldsymbol{C}^{(t)} = \begin{pmatrix} 1 & -\delta a & \delta p \\ \delta a & 1 & -\delta r \\ -\delta p & \delta r & 1 \end{pmatrix} = \boldsymbol{I} + \boldsymbol{\Omega}^{(t)}\delta t, \tag{15}$$

where

$$\boldsymbol{\Omega}^{(t)} = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}. \tag{16}$$

This allows us to relate the rotation matrix $\boldsymbol{R}^{(t)}$ with the rotation matrix of the next sampling period $\boldsymbol{R}^{(t+\delta t)}$. Then

$$\boldsymbol{R}^{(t+\delta t)} = \boldsymbol{R}^{(t)} \times \boldsymbol{C}^{(t)}, \tag{17}$$

where we have overloaded the superscript to mean the current sampling period besides the time slot. This gives

$$\frac{d\boldsymbol{R}^{(t)}}{dt} = \boldsymbol{R}^{(t)} \times \boldsymbol{\Omega}, \tag{18}$$

$$\boldsymbol{R}^{(t+\delta t)} = \boldsymbol{R}^{(t)} \times \exp\left(\int_t^{t+\delta t} \boldsymbol{\Omega} dt\right). \tag{19}$$

The DCM update equation is obtained as each new angular velocity samples comes by

$$\boldsymbol{R}^{(t+\delta t)} = \boldsymbol{R}^{(t)}\left(\boldsymbol{I} + \frac{\sin(\|\boldsymbol{\omega}\|\delta t)}{\|\boldsymbol{\omega}\|}\boldsymbol{\Omega} + \frac{1-\cos(\|\boldsymbol{\omega}\|\delta t)}{\|\boldsymbol{\omega}\|^2}\boldsymbol{\Omega}^2\right). \tag{20}$$

With the DCM updated, at each sample, the accelerometer data can easily be mapped from LCS to GCS by

$$\boldsymbol{a}^{(G,t)} = \boldsymbol{R}^{(t)} \cdot \boldsymbol{a}^{(t)}, \tag{21}$$

where G specifies that the vector is the GCS. Finally, the updated velocity vector is given by

$$\begin{bmatrix} v_n^{(t+1)} \\ v_e^{(t+1)} \\ v_d^{(t+1)} \end{bmatrix} = \begin{bmatrix} v_n^{(t)} \\ v_e^{(t)} \\ v_d^{(t)} \end{bmatrix} + \begin{bmatrix} a_x^{(G,t)} - n_{ax}^{(t)} \\ a_y^{(G,t)} - n_{ay}^{(t)} \\ a_z^{(G,t)} - n_{az}^{(t)} - g \end{bmatrix} \delta t \tag{22}$$

and the corresponding displacement vector

$$\begin{bmatrix} d_x^{(t+1)} \\ d_y^{(t+1)} \\ d_z^{(t+1)} \end{bmatrix} = \begin{bmatrix} v_n^{(t)}\delta t \\ v_e^{(t)}\delta t \\ -v_d^{(t)}\delta t \end{bmatrix} \tag{23}$$

$$\boldsymbol{R}^{(t)} = \begin{pmatrix} \cos(p)\cos(a) & -\cos(r)\sin(a) + \sin(r)\sin(p)\cos(a) & \sin(r)\sin(a) + \cos(r)\sin(p)\cos(a) \\ \cos(r)\sin(a) & \cos(r)\cos(a) + \sin(r)\sin(p)\sin(a) & -\sin(r)\cos(a) + \cos(r)\sin(p)\sin(a) \\ -\sin(p) & \sin(r)\cos(p) & \cos(r)\cos(p), \end{pmatrix} \tag{14}$$

is used in [12, Algorithm 3] to obtain the particle $x_i^{(t+1,l)}$.

4.4 Static Phase

During the static phase data from the accelerometer and the magnetometer are used to derive the pitch, roll, and attitude required for the rotation matrix, using (14). To compensate the tilt errors the following algorithm is used as presented in [9].

First, a linear-phase finite impulse response (FIR) low pass filter (LPF) is used to filter the accelerometer signal. The LPF is designed with a cutoff frequency of less than 1Hz, as a typical human stride takes $\approx$ 1s. The filtered acceleration $g^{(L)}$ is then normalized and redefined as a gravity vector in LCS. The normalized GCS gravity vector is then given by

$$\boldsymbol{g}^{(G)} = \boldsymbol{R} \cdot \boldsymbol{g}^{(L)}, \tag{24}$$

where $g^{(G)} = [0, 0, 1]^T$.

Solving the above equation for roll and pitch gives

$$p^{(t)} = \text{atan2}\left(\mathrm{g_x^{(t)}}, \sqrt{(\mathrm{g_y^{(t)}})^2 + (\mathrm{g_z^{(t)}})^2}\right), \tag{25a}$$

$$r^{(t)} = \text{atan2}\left(\mathrm{g_y^{(t)}}\text{sign}(\cos(\mathrm{p^{(t)}})), \mathrm{g_z^{(t)}}\text{sign}(\cos(\mathrm{p^{(t)}}))\right). \tag{25b}$$

After both pitch and roll have been found from the acceleration data, the attitude can be calculated from the magnetic field data. Let $\boldsymbol{\mu}^{(L,t)} = [\mu_x^{(t)}, \mu_y^{(t)}, \mu_z^{(t)}]$ be the LCS magnetometer readings. Then the compensated magnetic field can be calculated as

$$h_x^{(t)} = \mu_x^{(t)}\cos(p^{(t)}) + \mu_y^{(t)}\sin(p^{(t)})\sin(r^{(t)}) + \mu_z^{(t)}\sin(p^{(t)})\cos(r^{(t)}) \tag{25c}$$

$$h_y^{(t)} = \mu_y^{(t)}\cos(r^{(t)}) - \mu_z^{(t)}\sin(r^{(t)}) \tag{25d}$$

$$a^{(t)} = \text{atan2}(-\mathrm{h_y^{(t)}}, \mathrm{h_x^{(t)}}) - \mathrm{D}, \tag{25e}$$

where D is the magnetic declination, or the difference between the magnetic north and the true north, caused by the tilt of the earth magnetic field generator relative to the earth spin axis.

The algorithm is summarized as Algorithm 2.

4.5 Complexity

For the BP time slots, the complexity is due to the message passing algorithm used. In our case, as GBP is a parametric form message passing algorithm, the computational cost is $\mathcal{O}(\mathcal{N}_i L)$ [12]. This makes the algorithm an order of magnitude faster than non-parametric BP algorithms, e.g., SPAWN [10]. We also compare GBP with the SnG algorithm [11], which has a complexity of $\mathcal{O}(\bar{\mathcal{N}}_i L)$, where $\bar{\mathcal{N}}_i$ symbolizes the average pseudoanchors of node i. The number of particles used in both algorithms is approximately

Algorithm 2 PHIMTA

Sample $\{\pi_i^{(t,l)}, x_i^{(t,l)}\}_{l=1}^L \sim p(X_i^{(t)})$
Detect Stride Phase using SHOE Algorithm
if Stride Phase is *Stance* **then**
  Set $\boldsymbol{\omega}^{(t)} = 0$
  Set $\boldsymbol{v}^{(t)} = 0$
  Extract g from a using LPF
  Calculate p, r, a using equations (25)
  Calculate Rotation Matrix $\boldsymbol{R}^{(t)}$ using (14)
else
  sample $\{\boldsymbol{n}_\omega^{(l,t)}\}_{l=1}^L \sim \mathcal{N}(\boldsymbol{n}_\omega)$
  For each sample calculate $\boldsymbol{R}^{(t)}$ using (20)
end if
Sample $\{\boldsymbol{n}_a^{(l,t)}\}_{l=1}^L \sim \mathcal{N}(\boldsymbol{n}_a)$
Sample $\{\boldsymbol{n}_\mu^{(l,t)}\}_{l=1}^L \sim \mathcal{N}(\boldsymbol{n}_\mu)$
for each sample calculate $\{a^{(l,G,t)}\}_{l=1}^L$ using (21)
Calculate $\{\boldsymbol{x}^{(l,t+1)}\}_{l=1}^L$ using (23) and [12, Algorithm 3]
Convert to parametric form using [12, Algorithm 4]
Update belief $p(X_i^{(t+1)})$

$L = 100$, while the number of average pseudo-anchors will be less or equal to the average number of neighbors. As such, the algorithms tend to have similar complexity with SnG being slightly faster. Even though the two algorithms seem to have the same computational cost, a step by step comparsion in Table 1 clearly shows that GBP is faster, as it has fewer steps and there is no need to optimize the objective function at every iteration.

Table 1: Complexity of GBP vs SnG

GBP [12]		SnG [11]	
Step	Complexity	Step	Complexity
sample incoming message	L	sample incoming message	L
count sample IDS	|ID|	count-sort likelihoods	$\mathcal{N}_i L$
multiply multinomial pdfs	N_i|ID|	sort candidate points	$U\log(U)$
filter IDs	|ID| log(|ID|)	get centroid	$U_{\max}\log(U_{\max})$
—	—-	IWLS	$\mathcal{N}_i I_{\text{IWLS}}$

We assume that both algorithms approximately use the same number of particles, and |ID| is the cardinality of relevant IDs in GBP, while U and $U_{\max}$ is the number of candidate points and number of highest likelihood candidate points respectively. Finally I_{IWLS} is the number of iterations IWLS can run.

For the IMU time slots, the complexity is due to PHIMTA.

All the steps are proportional to either the number of signals Y obtained from MEMS, or the number of particles used in the calculations L. Each step is given with the corresponding cost in Table 2. Hence the complexity is $\mathcal{O}(YL)$. We compare PHIMTA with the PDR algorithm in [18]. Even if the complexity scales in the same way, PDR has fewer computations per iteration than the hybrid algorithm. Essentialy it is a compromise between computational cost, and accuracy as will be seen in the sequel. By using GBP though, the computational increase from PHIMTA can be easily compensated.

Table 2: Complexity of PHIMTA

Step	Complexity
sample $p(X_i^{(t)})$	L
SHOE Algorithm	Y
Stride phase	YL
Quasi-static phase	YL
update position	L
convert to parametric form	L

Table 3: Complexity costs of GBP/PHIMTA vs SnG/PDR

Algorithm	Complexity
Grid-BP	$\mathcal{O}(\mathcal{N}_i L_{\text{GBP}})$
PHIMTA	$\mathcal{O}(YL_{\text{PHIMTA}})$
SnG	$\mathcal{O}(\mathcal{N}_i L_{\text{SNG}})$
PDR	$\mathcal{O}(YL_{\text{PDR}})$

Note that $L_{\text{GBP}} = L_{\text{PHIMTA}} = L_{\text{SNG}} = L_{\text{PDR}} \simeq 100$.

5. SIMULATION RESULTS

Our proposed algorithm was evaluated using Monte-Carlo simulations. We considered a 2D grid 20m $\times$ 20m with 4 anchors at the corners of the grid. Ten nodes are randomly placed inside the grid and are trying to localize. We first consider a static scenario in which the root-mean-square (RMS) localization error is compared between GBP and SnG and NBP [19], as an implementation of the SPAWN framework.

In Fig. 3, we illustrate the average RMS error for various communication ranges. We assume that the number of particles used is 300 and $K_e = 0.001$. As expected NBP outperforms both SnG and GBP but at a greater computational cost. While GBP provides similar RMS to SnG for lower communication range, it is interesting to note that both NBP and GBP take advantage of the availability of more neighbors while SnG seems to keep a constant RMS error. We should also mention that SnG would collapse if the average number of neighbors is too low. Finally, for the SnG simulations, we initially ran cooperative least-square (LS), cf. [10], as it requires initial estimates to run, which is not required for NBP and GBP.

Now, we consider a mobility scenario where the 10 nodes move randomly for a period of 180s. Simulation parameters are the same as before with a communication range of 12m. We used the test data in [13, 14], as a pool of possible movements that a node can follow with the respective

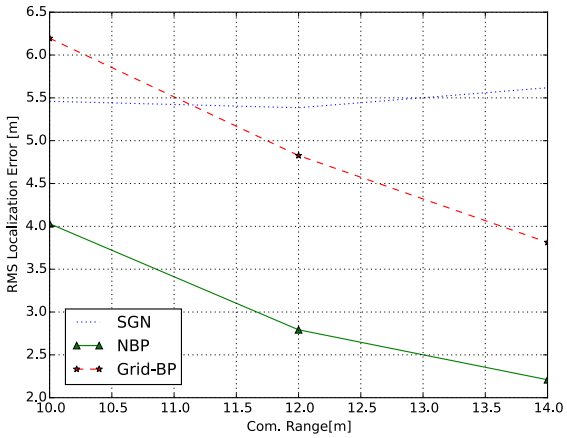

Figure 3: The average RMS error versus the communication range.

MEMS measurements. Each node decides by a stationary probability $\Pr_i(s)$ if it will wait or not and for how many seconds. If it will wait, then it will be used in the cooperative localization algorithm. Alternatively, it will pick randomly a movement from the ones provided in the test data and move accordingly until the movement time elapses and the procedure repeats until the simulations finish. The cooperative localization steps occur, to obtain starting locations and afterwards every 10s. In Fig. 4, we present the average RMS tracking error results, which is the average RMS localization error per second for the network. Obviously, as the stationary probability increases, the nodes move less and consequently are more readily available in the cooperative localization steps, improving their RMS tracking error. At the boundary scenario, we have $\Pr_i(s) = 0$ which means that all nodes are constantly moving and hence besides the initial cooperative localization step, nodes will only use MEMS information algorithms. As we can see, the superiority of PHIMTA over PDR is evident in all mobility scenarios. Secondly, SnG slightly outperforms GBP. This is due to the ability of SnG to filter out nodes that mistakenly believe they are stationary while in reality their are moving. Despite that, the difference is small, given the drastic decrease in computational cost. Finally it is clear that using cooperative localization is a great addition to MEMS localization.

6. CONCLUSION

This paper presented a novel hybrid approach to pedestrian mobility tracking using a hybrid cooperative localization and MEMS tracking approach. The end result is a powerful and promising mobility tracking algorithm with very low computational requirements compared to the literature.

7. REFERENCES

[1] S. Rezaei and R. Sengupta, "Kalman filter-based integration of DGPS and vehicle sensors for

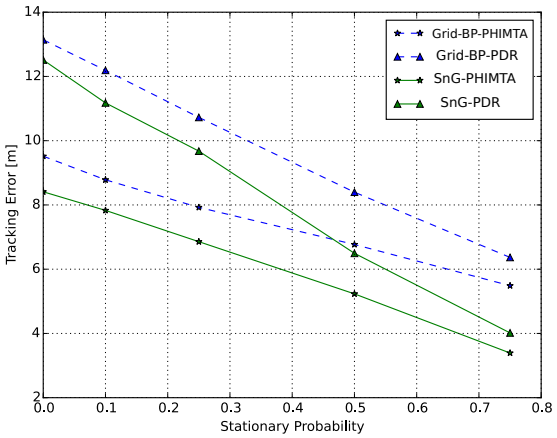

Figure 4: The average RMS tracking error with $K_e = 0.01$.

localization," *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 6, pp. 1080–1088, Nov. 2007.

[2] R. Zekavat and R. M. Buehrer, *Handbook of position location: Theory, practice and advances*. John Wiley & Sons, 2011, vol. 27.

[3] J. Georgy, A. Noureldin, and M. Bayoumi, "Mixture particle filter for low cost INS/odometer/GPS integration in land vehicles," in Proc. *IEEE Veh. Technol. Conf. (VTC Spring 2009)*, pp. 1–5, 26-29 Apr. 2009, Barcelona, Spain.

[4] W. Kang, S. Nam, Y. Han, and S. Lee, "Improved heading estimation for smartphone-based indoor positioning systems," in Proc. *IEEE Pers. Indoor and Mobile Radio Commun. (PIMRC)*, pp. 2449–2453, 9-12 Sep. 2014, Sydney, Australia.

[5] Z. Sun, X. Mao, W. Tian, and X. Zhang, "Activity classification and dead reckoning for pedestrian navigation with wearable sensors," *Meas. Sci. Technol.*, vol. 20, no. 1, 2009.

[6] K. Altun and B. Barshan, "Pedestrian dead reckoning employing simultaneous activity recognition cues," *Meas. Sci. Technol.*, vol. 23, no. 2, 2012.

[7] A. G. Quinchia, C. Ferrer, G. Falco, E. Falletti, and F. Dovis, "Analysis and modelling of MEMS inertial measurement unit," in Proc. *Int. Conf. Localization and GNSS*, pp. 1–7, 25-27 Jun. 2012, Starnberg.

[8] N. El-Sheimy, H. Hou, and X. Niu, "Analysis and modeling of inertial sensors using Allan variance," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 1, pp. 140–149, Jan. 2008.

[9] C. Huang, Z. Liao, and L. Zhao, "Synergism of INS and PDR in self-contained pedestrian tracking with a miniature sensor module," *IEEE Sensors J.*, vol. 10, no. 8, pp. 1349–1359, Aug. 2010.

[10] H. Wymeersch, J. Lien, and M. Win, "Cooperative localization in wireless networks," *Proc. IEEE*, vol. 97, no. 2, pp. 427–450, Feb. 2009.

[11] T. Higuchi, S. Fujii, H. Yamaguchi, and T. Higashino, "Mobile node localization focusing on stop-and-go behavior of indoor pedestrians," *IEEE Trans. Mobile Comput.*, vol. 13, no. 7, pp. 1564–1578, Jul. 2014.

[12] P. A. Oikonomou-Filandras, K. K. Wong and Y. Zhang, "Grid-based belief propagation for cooperative localization," arXiv, 1509.03287 2015 [online] Available: http://arxiv.org/abs/1509.03287

[13] M. Angermann, A. Friese, and M. Khider, "A reference measurement data set for multisensor pedestrian navigation with accurate ground truth," *European Navigation Conf.*, May 2009, Naples, Italy.

[14] M. Angermann, P. Robertson, T. Kemptner, and M. Khider, "A high precision reference data set for pedestrian navigation using foot-mounted inertial sensors," in Proc. *Int. Conf. Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–6, 15-17 Sep. 2010, Zurich.

[15] R. Buehrer, T. Jia, and B. Thompson, "Cooperative indoor position location using the parallel projection method," in Proc. *Int. Conf. Indoor Positioning Indoor Navigation (IPIN)*, pp. 1–10, 15-17 Sep. 2010, Zurich.

[16] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. Cambridge, MA : MIT Press, 2009.

[17] I. Skog, P. Handel, J. O. Nilsson, and J. Rantakokko, "Zero-velocity detection–An algorithm evaluation," *IEEE Trans. Biomed. Eng.*, vol. 57, no. 11, pp. 2657-2666, Nov. 2010.

[18] Y. Jin, H.-S. Toh, W.-S. Soh, and W.-C. Wong, "A robust dead-reckoning pedestrian tracking system with low cost sensors," in Proc. *Int. Conf. Pervasive Comput. Commun. (PerCom)*, pp. 222–230, 21-25 Mar. 2011, Seattle, USA.

[19] A. Ihler, J. Fisher, R. Moses, and A. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 4, pp. 809–819, Apr. 2005.