Smartphone-based Real-time Indoor Location Tracking with One-meter Precision

Po-Chou Liang and Paul Krause, University of Surrey

Abstract—Monitoring the activities of daily living of the elderly at home is widely recognized as useful for the detection of new or deteriorating health conditions. However, the accuracy of existing indoor location tracking systems remains unsatisfactory. The aim of this study was therefore to develop a localization system that can identify a patient’s real-time location in a home environment with maximum estimation error of two meters at a 95% confidence level. A proof-of-concept system based on a sensor fusion approach was built with considerations for lower cost, reduced intrusiveness, and higher mobility, deployability, and portability. This involved the development of both a step detector using the accelerometer and compass of an iPhone 5 and a radio-based localization subsystem using a Kalman filter and received signal strength indication (RSSI) to tackle issues that had been identified as limiting accuracy. The results of our experiments were promising with an average estimation error of 0.47 meters. We are confident that with the proposed future work our design can be adapted to a home-like environment with a more robust localization solution.

Index Terms—telemonitoring, localization, received signal strength, Kalman filter, step detection, sensor fusion

I. INTRODUCTION

For years, there has been a lack of robust evidence for cost-effectiveness of remote healthcare systems [1,2]. To address this issue, we set up as our core objective the cost-effective design of real-time home healthcare telemonitoring. Key considerations about the development of our system included lower cost, reduced intrusiveness, and higher mobility, deployability, and portability. A prototype, consisting of vital sign, safety, and location tracking, was then developed [3]. The purpose of this paper is to present our proposed real-time indoor location tracking mechanism on the client side, composed of a User Agent module (based on iPhone 5) and a number of low-cost sensors.

In healthcare, it has been widely acknowledged that in-home monitoring of the elderly or chronic disease outpatients’ daily movement patterns is useful for detection of early signs of new or deteriorating health issues. However, to achieve satisfactory accuracy remains a challenge for indoor location tracking [4]. For the purpose of this study, we set up our basic requirement as to identify the patient’s real-time location within a home environment with maximum estimation error of two meters at a 95% confidence level.

A radio-based localization technique using trilateration, trigonometry and received signal strength indication (RSSI) from three triangular deployed Bluetooth Low Energy (BLE) sensors was chosen to build the required functionality. Meanwhile, to tackle the problem of arbitrary variations of RSSI readings in an indoor environment and to enhance the overall reliability of our system, we developed the three following elements: a step detection mechanism to produce extra patient location information; a discrete-time Kalman filter to improve distance estimation; and a tight coupling sensor fusion approach to integrate all these features as a whole.

The results of our 24 experiments, each performed by a user carrying a User Agent and walking around a small office (54 square meters) for up to 65 seconds, were promising with an average estimation error of 0.47 meters. To illustrate both the issues encountered and our solutions to those issues, the remainder of this paper is structured as follows. In section II, we briefly introduce related work in indoor location tracking and the concepts of RSSI and Kalman filter. In Section III, we describe our design and implementation of our proposed system. Section IV presents the experimental results and our evaluation. Finally, Section V provides our conclusions and suggestions for future work. A preliminary version of this work has been reported [14]. This paper expands significantly on that presentation in both technical details and extent of the evaluation.

II. RELATED WORK

A. Step Detection

As stated by [15], most studies on step detection have emphasized estimation of step counts, for example [16,17], rather than indoor location tracking. However, for indoor localization based on step detection, [15] used triads of accelerometers, angular rate sensors, and magnetometers (sampling rate ≈ 70 Hz) attached to the instep of a user to estimate the velocity and distance of movement and detect gait events. The average distance estimation error for their indoor 16-step straight-line walking experiments was 5.5% with a maximum error of 2.05 meters.

[18,19] used smartphones’ built-in inertial sensors (based on <25 Hz and 50 Hz sampling rate, respectively) to detect steps
and heading direction in order to estimate user location. Together with a floor map, [18] in their hand-held experiments achieved a mean error of 1.5 meters and a 95th percentile error of 4.3 meters. [19] required using two sets of sensors (i.e. two smartphones) at one time, both placed in the user’s trousers pockets, to achieve an average estimation accuracy of about 1.6 meters in their 10 2-minute walking experiments.

In our opinion, the high sampling rate employed by these studies, though providing more precise measurements, would significantly drain a smartphone’s battery. Besides, both mounting sensors to the instep and using two smartphones at one time were either intrusive or impractical.

B. Indoor Radio-based Location Tracking

For indoor radio-based location tracking, there are a number of techniques, including RSSI, angle of arrival (AOA), time of arrival (TOA), and time difference of arrival (TDOA). The first one estimates the distance directly from the strength of the received signal, whereas AOA is based on both the signal strength and angle of each beam from a usually costly antenna array. The remaining two basically convert the travel time of signals into distances. We chose RSSI to build our system because AOA is more costly, and both TOA and TDOA require a very high resolution of timer, which is generally not available on a commodity device, such as a smartphone. Though all these techniques have gained popularity in recent years, some major problems in this field, as indicated by [4], remain unresolved, such as computationally intensive algorithms, excessive access point installations and unstable wireless signal transmission.

To improve distance estimates, both the Monte Carlo [6,8] and Kalman filter [7,9,10] methods were commonly adopted for constructing RSSI-based localization algorithms. This is because both can produce statistically more precise estimation of system states than those solely based on one or a few noisy measurements. However, from our perspective, the need for an extra computer to perform location estimates using the Monte Carlo method, as proposed by some studies [6,8], raises cost and reliability issues.

Another common technique was to produce and store a detailed wireless signal strength map at each specific survey location before performing localization [4,8,9]. Location tracking was then conducted by comparing real-time RSSI measurements with the stored signal strength map. However, we consider that the need to produce such a map for each survey location would cause issues on system deployability and portability, as well as on longer processing time and higher system requirements, such as storage and database. In some cases, excessive sensor nodes were needed to conduct target tracking. For instance, [7] used both eight static sensor nodes and one or more mobile nodes in its simulations.

With regard to accuracy and performance, some studies could achieve an overall estimation error of about two meters. For example, the distance estimation errors of [4] implemented in an iPhone were within 2.3 meters with 90% precision, whereas the average error for walk-through tests of [8] was 2.1 meters. By applying a modified extended Kalman filter on an existing RSSI data set, [10] achieved an average estimation error of 2.11 meters. Though the overall distance estimate error of [6] was 1.2 meters, its maximum error was about 2.5 meters and the use of a networked computer for performing heavy computational tasks caused latency of up to eight seconds. [11] claimed that they had estimation errors of below micrometers in their software simulations by using TOA based trilateration with IEEE 802.11v. We argue that this can neither be applied to current smartphones, nor to commodity desktop systems, because these systems’ timer resolutions would have needed to be up to 3.33 femtoseconds (i.e. 10^-15) to achieve a distance precision in micrometers. (According to Microsoft [12], the maximum timer resolution in Windows is 1 millisecond.)

The identified issues mentioned above indicate a remaining gap in achieving wider uptake of indoor location tracking in society. Moreover, lessons learned from these studies suggest that we need to develop a lightweight, but accurate localization algorithm suitable for execution in a smartphone. We should also avoid adopting detailed wireless signal maps, as well as excessive hardware installations. These would help achieve higher mobility, reliability, deployability and portability of our real-time telemonitoring system at a lower cost.

C. Received Signal Strength Indication

Theoretically, this indication is based on the inverse-square law that the wireless signal strength is proportional to the inverse of the square of the distance from the signal source. Equation (1) denotes the relationship between received signal strength and corresponding distance [5]. However, in reality, due to several issues, such as multi-path fading, indoor shadowing, and interference, the relation between signal strength and distance in an indoor environment usually involves a much wider range of factors. This significantly increases the complexity of RSSI-based distance estimates. To improve the accuracy of RSSI-based location tracking, numerous approaches and algorithms, such as those mentioned in the previous sub-section, have been proposed.

\[ \text{RSSI}_d = -10 \log(d) + \text{RSSI}_0 \]  
(1)

In (1), RSSI_d (in Decibel-milli watts) is the RSSI measured at distance \( d \) (in meters) from the source; \( n \) is the path loss exponent and RSSI_0 (in Decibel-milli watts) is the RSSI measured at one-meter distance. Further details about how (1) is derived is outside the scope of this paper, but can be found in a number of foundational texts, such as [5].

D. Kalman Filter

Because a Kalman filter is relatively lightweight and has a much better convergence rate than a Monte Carlo filter, we chose the former to build our localization algorithm. Equations (2) and (3) explain how to model a system using discrete-time Kalman filter [13]:

\[ x_k = Ax_{k-1} + Bu_k + w_k \]  
(2)

\[ z_k = Hx_k + v_k \]  
(3)

In (2), \( x_k \) is the estimate of the system state variable at time \( k \); \( u_k \) is the control signal; and \( w_k \) is the process noise. In (3), \( z_k \) is the measurement value at time \( k \); and \( v_k \) is the measurement...
noise. A, B and H are general form matrices, introduced to model the system, and in many cases can be simplified as numeric values. To perform the estimates based on a discrete-time Kalman filter, the two sets of equations in Table I can be used repeatedly.

### Table I. Two Sets of Discrete-time Kalman Filter Equations for State Estimation, after [13]

<table>
<thead>
<tr>
<th>Time Update (Prediction) Equations</th>
<th>Measurement Update (Correction) Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_k = A_{k-1} x_{k-1} + B u_{k-1} + w_{k-1} )</td>
<td>( K_k = P_k H^T (H P_k H^T + R)^{-1} )</td>
</tr>
<tr>
<td>( P_k = A P_{k-1} A^T + Q )</td>
<td>( x_k = x_k + K_k (z_k - H x_k) )</td>
</tr>
<tr>
<td>( P_k = (I - K_k H) P_k )</td>
<td>( P_k )</td>
</tr>
</tbody>
</table>

In Table I, \( x_k \) is the “a priori” estimate of the system state at time \( k \) before measurement update correction; \( x_{k-1} \) is the updated estimate (or a posteriori estimate) at time \( k-1 \) after measurement; \( P_k \) is the “a priori” error covariance; \( R \) is the measurement error covariance; \( K_k \) is the Kalman Gain; \( x_k \), which is the updated estimate (or a posteriori estimate) at time \( k \) after measurement, is the very value we try to find; and \( P_k \) is the updated error covariance after measurement. More details about how these parameters are applied can be found in existing literature on the topic, such as [13].

### III. Design and Implementation

Our designs and experiments started from the use of raw RSSI data received from three triangular deployed BLE sensors (i.e. TI SensorTags) to estimate the location and movement of a tracked target who held a User Agent (i.e. an iPhone 5) in their hand and walked around a small concrete-walled office (9 meters × 6 meters). The three BLE sensors (denoted as \( S_0 \), \( S_1 \), and \( S_2 \)) were placed against three different walls of the office at 1.1 meter height from the floor. However, because of the very diverse and unstable RSSI signals for the reasons discussed above, it was impossible to use the raw RSSI data alone to reliably estimate the location and movement of the target.

### Table II. Raw RSSIs Recorded at Different Distances from Sensor \( S_i \)

<table>
<thead>
<tr>
<th>Distance (meter)</th>
<th>Min. RSSI (dBm)</th>
<th>Max. RSSI (dBm)</th>
<th>Mean RSSI (dBm)</th>
<th>Standard deviation (dBm)</th>
<th>Path loss exponent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-70</td>
<td>-63</td>
<td>-66.9</td>
<td>1.87</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>-87</td>
<td>-66</td>
<td>-73</td>
<td>5.83</td>
<td>NA</td>
</tr>
<tr>
<td>2</td>
<td>-90</td>
<td>-72</td>
<td>-78.7</td>
<td>4.63</td>
<td>1.93</td>
</tr>
<tr>
<td></td>
<td>-94</td>
<td>-67</td>
<td>-73.7</td>
<td>5.57</td>
<td>2.24</td>
</tr>
<tr>
<td>4</td>
<td>-107</td>
<td>-77</td>
<td>-85.9</td>
<td>6.98</td>
<td>2.16</td>
</tr>
<tr>
<td></td>
<td>-74</td>
<td>-69</td>
<td>-71.1</td>
<td>1.41</td>
<td>0.69</td>
</tr>
<tr>
<td>6</td>
<td>-103</td>
<td>-87</td>
<td>-94.6</td>
<td>4.27</td>
<td>2.79</td>
</tr>
<tr>
<td></td>
<td>-80</td>
<td>-73</td>
<td>-76.1</td>
<td>1.9</td>
<td>1.17</td>
</tr>
</tbody>
</table>

Table II shows the results of two 10-minute raw RSSI measurements recorded at a distance of one to six meters from sensor \( S_i \). For example, during the first recording the derived mean RSSI at one-meter distance from sensor \( S_1 \) was -66.9 dBm with a standard deviation of 1.87 dBm, whereas the mean decreased to -73 dBm with a standard deviation of 5.83 during the second recording. The values of the path loss exponent at different distances were estimated by using (1).

We then implemented a Kalman filter to improve RSSI estimates based on noisy RSSI measurements. However, some identified issues in relation to the one-second RSSI update rate implemented by iOS caused the resultant RSSI estimates to become very unreliable. Consequently, we decided to take a sensor fusion approach to perform location estimates. Based on this approach, the design and implementation of a step detector, a Kalman filter, and several estimate optimization mechanisms are described in the following.

### A. Step Detector

A step detector was designed and implemented on the User Agent using acceleration parameters (generated by the iPhone’s built-in accelerometer) and heading information (by compass) at a 10 Hz sampling rate. Table III summarizes the algorithm. Based on empirical experiments and signal analysis, two thresholds, i.e. 1.07 gravities (\( SV_{THRESHOLD1} \)) followed by a reduction of 0.209 gravities (\( SV_{THRESHOLD2} \)) of the sum vector (SV) of acceleration in X-Y-Z axes within 400 milliseconds, were defined. If these thresholds were met during real-time monitoring, the User Agent would signify a detected step when, at the same time, the forward acceleration was less than or equal to -0.3 gravities (\( FWD_{ACC_{THRESHOLD}} \)) or the change of heading angle within 100 milliseconds was less than seven degrees (\( HEADING_{THRESHOLD} \)). If two consecutive steps were signified within 200 milliseconds, the second one was regarded as faulty and hence was discarded.

### Table III. Step Detection Algorithm

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( SV \gets \text{getLatestSV}(); )</td>
</tr>
<tr>
<td>2</td>
<td>if ( \text{firstFlagSV} == 0 ) then</td>
</tr>
<tr>
<td>3</td>
<td>if ( SV &gt; SV_{THRESHOLD1} ) then</td>
</tr>
<tr>
<td>4</td>
<td>( \text{firstFlagSV} \leftarrow \text{SV}; \text{fwdAcc} \leftarrow \text{getForwardACC}(); )</td>
</tr>
<tr>
<td>5</td>
<td>end</td>
</tr>
<tr>
<td>6</td>
<td>else if ( \text{sndFlagSV} == 0 ) and ( \text{timeFromFirstFlagSV}() \leq 400\text{ms} ) then</td>
</tr>
<tr>
<td>7</td>
<td>if ( SV &gt; \text{firstFlagSV} ) then</td>
</tr>
<tr>
<td>8</td>
<td>( \text{firstFlagSV} \leftarrow \text{SV}; \text{fwdAcc} \leftarrow \text{getForwardACC}(); )</td>
</tr>
<tr>
<td>9</td>
<td>else if (( \text{firstFlagSV} - \text{SV} )) ( &gt; \text{SV}_{THRESHOLD2} ) then</td>
</tr>
<tr>
<td>10</td>
<td>( \text{sndFlagSV} \leftarrow \text{SV}; )</td>
</tr>
<tr>
<td>11</td>
<td>if (( \text{getHeadingChangeIn100ms}() \geq \text{HEADING}<em>{THRESHOLD} ) and ( \text{fwdAcc} &gt; \text{FWD}</em>{ACC_{THRESHOLD}} )) or ( \text{timeFromPreviousStep}() \leq 200\text{ms} ) then</td>
</tr>
<tr>
<td>12</td>
<td>( \text{resetAllFlags}(); )</td>
</tr>
<tr>
<td>13</td>
<td>else</td>
</tr>
<tr>
<td>14</td>
<td>( \text{signifyOneStepIsDetected}(); )</td>
</tr>
<tr>
<td>15</td>
<td>end</td>
</tr>
<tr>
<td>16</td>
<td>end</td>
</tr>
<tr>
<td>17</td>
<td>end</td>
</tr>
<tr>
<td>18</td>
<td>( \text{resetAllFlags}(); )</td>
</tr>
<tr>
<td>19</td>
<td>End</td>
</tr>
</tbody>
</table>

Whenever a step or several steps were detected before the User Agent received new RSSI updates from all three reference sensors, a new location of the target, as well as the change of distance between the target and each of the sensors during this period of time, was calculated based on both the pre-defined length of a step (e.g. 0.65 meters) and the heading data.

Fig. 1 gives two scenarios in which the target \( T \) is moving from one known location, denoted as \( T_{old} \) (\( x_i, y_i \)), to another unknown location, denoted as \( T_{new} \) (\( x_i + \Delta x, y_i + \Delta y \)) for one step detected or \( T_{new} \) (\( x_i + \Delta x, y_i + \Delta y \)) for two steps detected before the User Agent receives new RSSI updates. Here (\( x_i, y_i \)) is the coordinate of \( T_{old} \) and \( S_i \) refers to the \( i \)th sensor placed at location (\( x_i, y_i \)). The distances between \( S_i \) and \( T_{old} \) and between \( S_i \) and \( T_{new} \) (or \( T_{new}^1 \)) are indicated as \( d_{i,old} \).
and $d_{i,new1}$ (or $d_{i,new2}$) respectively and the length of each step of the target (or patient) is $\lambda$. The $North_{mag}$ represents the direction of magnetic north with its heading angle at 0°, while $X_{env}$ and $Y_{env}$ refer to the X- and Y-axis, respectively, in the monitoring environment. $Y_{env}$ is pointing toward $\theta$, $X_{env}$ is pointing toward $\theta + 90^\circ$, and the target is heading toward $h_i$ in Fig. 1(a) and $h_2$ in Fig. 1(b) respectively. For the scenario in Fig. 1(a), we can calculate the displacement of the target in both X- and Y-axis and the change of distance (denoted as $d_{i,chg1}$) between the target and sensor $S_i$ using (4), (5), and (6), as follows:

$$\phi_i = ((\theta + 90') \bmod (360')) - h_i$$

(4)

$$\Delta x_i = \lambda \times \cos(\phi_i) \text{ and } \Delta y_i = \lambda \times \sin(\phi_i)$$

(5)

$$d_{i,chg1} = d_{i,new1} - d_{i,old}$$

$$= \sqrt{(x_i + \Delta x_i - x_j)^2 + (y_i + \Delta y_i - y_j)^2} - \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

(6)

For the scenario in Fig. 1(b), we can calculate $\phi_2$, $\Delta x_2$, $\Delta y_2$, and $d_{i,new2}$ in a similar way and then work out the displacement of the target and the change of distance (denoted as $d_{i,chg2}$) between the target and sensor $S_i$ using (7):

$$d_{i,chg2} = d_{i,new2} - d_{i,old}$$

(7)

### B. Kalman Filter for Distance Estimates

A discrete-time Kalman filter was developed for estimating the distance (denoted as $d_{i,k}$; hereafter an added subscript $i$ to each variable shown in Table I refers to sensor $S_i$) between the target and sensor $S_i$ (subscript $i$ could be 0, 1 or 2) at time $k$. Upon receiving new RSSI updates from a sensor, the distance information calculated by the step detector would be used as the control signal $u_{i,k}$ of the Kalman filter. The reason for waiting for new RSSI updates before starting a new run of state estimation via the Kalman filter was to synchronize the target movement with RSSI readings. This was because the iOS updates RSSI measurements for each sensor at a maximum rate of once per second and updates for different sensors usually occur at different times, whereas a walking step generally takes less than one second. This synchronization process would incur a latency of less than a second in most cases, but up to five seconds in the worst case.

When a step was correctly detected, the process noise, which occurred mainly due to the inaccuracy of inferring the heading angle and step length, would be relatively small. So we used a value between zero and 0.13 meters (i.e. the standard deviation of step detection, if we assumed that on average one in every twenty-five steps was incorrectly detected or missed) as the process noise $w_{i,k}$, depending on how much correlation between the step detector and the Kalman filter was needed. The process noise covariance $Q_i$ was equal to $E[w_{i,k}w_{i,k}^T]$, where $E[w_{i,k}]$ is the expected value of $w_{i,k}$ and $w_{i,k}^T$ is the transpose of $w_{i,k}$.

The value for measurement noise generally became bigger with the increase of distance between the User Agent and sensor $S_i$. Based on our empirical experiments in fine tuning different parameters to get the best estimation results, we assumed that the measurement noise $v_{i,k}$ was equal to the converted distance multiplied by a constant factor of 0.6. Meanwhile, matrices $A$, $B$ and $H$ in (2) and (3) were all simplified as a numerical constant of one and the path loss exponent $n$ in (1) was set to 2.4 for $S_0$, as well as 2.2 and 2.1 for $S_1$ and $S_2$ respectively. The initial value of $d_{i,0}$ was set to a pre-defined distance, as we assumed the real-time monitoring always started at a known location and the error covariance $P_{i,0}$ was simply set to a non-zero value, e.g. 0.5. With all these considerations and assumptions, the two sets of the Kalman filter equations as shown in Table I were ready for estimating the distance between the User Agent and each of the three sensors. Both a time update equation and a measurement update equation are rewritten using the aforementioned variables in the following:

$$d_{i,k} = d_{i,k-1} + d_{i,chg} + w_{i,k}$$

(8)

$$d_{i,k} = d_{i,k} + k_{i,k}(d_{i,max} - d_{i,k})$$

(9)

In (8), $d_{i,chg}$, which refers to the change of distance between target and sensor $S_i$ during the time period between RSSI updates received by the User Agent, is the control signal of our Kalman filter. In (9), $d_{i,max}$ is the distance converted from RSSI measurements received from sensor $S_i$.

### C. Mechanisms for Estimate Optimization

To calculate the real-time location of the tracked target based on the outputs from the Kalman filter, traditional techniques, including trilateration, trigonometric functions and Maximum-Likelihood Estimation (MLE) were first adopted. (The details of implementing MLE and trilateration are outside the scope of this paper.) To apply trilateration, the distances from three sensors estimated by the Kalman filter at a certain time were used as radii to create three circles each centered at $S_0$, $S_1$, and $S_2$ respectively. Ideally, there should have been a joint intersection point among the three circles, representing the optimally estimated real-time location of the target at that precise moment. However, the reality was much more complicated with both underestimates and overestimates of the three distances, probably happening at the same time.

Therefore, when there was an area of intersections, rather than a single point, found among the three circles, both trigonometric functions and an iterative procedure based on (10), the equation for MLE, were adopted to find an estimated target location $(x, y)$ within the intersections, with a least value of $\sigma_x$. In (10), $(x, y)$ refers to the coordinate of sensor $S_i$ and $d_i$ is the estimated distance from $S_i$ via Kalman filter.
Alternatively, if spheres were found only between two circles or there was not any intersection among the three circles, we progressively enlarged some or all of the circles by 0.05 meters at a time until intersections were found, and then we applied the techniques mentioned above. However, the resultant estimates were still not satisfactory.

\[
\sigma_{x,y} = \sum_{i=0}^{2} \sqrt{(x-x_i)^2 + (y-y_i)^2 - d_i^2}
\]  

(10)

Our assessment revealed that there might be two main problems that caused such unsatisfactory results. Firstly, the inaccuracy and insufficient number of RSSI measurements had inborn negative impacts on the accuracy of distance estimation. Secondly, the process of manipulating estimated distances to create joint intersections among the three circles had introduced further noise into the system.

Fig. 2 provides an example of how the radius manipulation process affects the final location estimation, when the estimated distances between the target and each of three or more sensors are not accurate. In Fig. 2(a), three estimated distances via Kalman filter at time \( k \) are denoted as \( d_{0,k} \), \( d_{1,k} \), and \( d_{2,k} \) and the actual location of the tracked target is marked with a grey “x” and labeled as \( T \). Meanwhile, \( T_{est}(x_{est},y_{est}) \) denotes the estimated location and its coordinate. Since there are no common intersections among the three circles, centered at \( S_0, S_1, \) and \( S_2 \) respectively, there is a possibility of underestimate of either \( d_{1,k} \) or \( d_{2,k} \), or of both. In order to apply location estimation mechanisms, we can enlarge either circle \( S_0 \) or \( S_1 \) by increasing \( d_{1,k} \) or \( d_{2,k} \) respectively, as depicted in Fig. 2(b) and 2(c).

![Fig. 2. Scenarios of radius manipulation process](Image 49x234 to 298x392)

In Fig. 2(b), the created intersection area (the arrow-pointing area) is further away from the actual target location, which would compromise the accuracy of \( T_{est}(x_{est},y_{est}) \). In comparison, the created intersection area in Fig. 2(c) is nearer to the actual location of the target. If we further enlarge circle \( S_2 \) a bit, the result would be even more accurate. However, as we do not possess the knowledge of the target’s actual location, it is not feasible to set a clear rule about whether we should enlarge circle \( S_1 \) or \( S_2 \) (or both) and to what extent we should adjust their radii (i.e. the estimated distances \( d_{i,k} \)) in such a situation.

It is apparent that one way to tackle the first identified problem is to take a large number of RSSI measurements; enough to produce both a detailed wireless signal strength map for each patient’s home and find out more appropriate values for relevant parameters prior to performing real-time monitoring. However, by doing so we would complicate the process of system deployment and set-up.

Consequently, the follow-step-detection mechanism was developed to address this problem by choosing a location \((x, y)\), nearest to the estimated target location \((x_e, y_e)\) based on step detection rather than MLE, within the intersection area. This means finding a location \((x, y)\) where \( \sigma_{x,y} \) in (11) has the least value.

\[
\sigma_{x,y} = \sqrt{(x-x_i)^2 + (y-y_i)^2}
\]  

(11)

Furthermore, to address these two issues simultaneously, another tight coupling mechanism was also developed by using the average of distances both estimated by step detection and converted from RSSI readings as the measurement value, e.g. \( d_{comm} \) in (9). When the difference between two consecutive RSSI readings was less than or equal to two (i.e. \( RSSI \_CHG\_THRESHOLD \)), meaning these RSSI measurements might be more accurate than usual, we halved the measurement noise factor for the next run of the Kalman filter. This mechanism then use (11) to choose one existing intersection point nearest to the target location estimated by step detection as the final estimated location of the target at that precise moment. Table IV summarizes this mechanism.

<table>
<thead>
<tr>
<th>Table IV. Tight-Coupling Sensor Fusion Mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>if hasReceivedNewRSSI== TRUE then</td>
</tr>
<tr>
<td>distanceByStep &lt;-- getNewDistanceFromStepDetection();</td>
</tr>
<tr>
<td>distanceByRSSI &lt;-- getNewDistanceFromRSSI();</td>
</tr>
<tr>
<td>rssChange &lt;-- abs(currentRSSI – previousRSSI);</td>
</tr>
<tr>
<td>if rssChange &lt;= RSSI_CHG_THRESHOLD then</td>
</tr>
<tr>
<td>measurementNnoiseFactor &lt;-- measurementNnoiseFactor/2;</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>measurementValue &lt;-- (distanceByStep + distanceByRSSI)/2;</td>
</tr>
<tr>
<td>newDistanceByKalman &lt;-- performKalmanMeasurementUpdate();</td>
</tr>
<tr>
<td>newLocation &lt;-- performFollowStepDetectionForNewLocation();</td>
</tr>
</tbody>
</table>

IV. Experiment Results and Evaluations

24 trials were performed by a user holding a User Agent in their hand with the screen facing upward (at an elevation angle of approximately 25 degrees) and walking around a concrete-walled office (9 meters × 6 meters) for up to 65 seconds. The four mechanisms mentioned in the previous section i.e. step detection, MLE, follow-step-detection, and tight-coupling sensor fusion, were implemented to track the user’s real-time movement. Fig. 3 shows the one-minute tracked paths based on the step detection and tight-coupling sensor fusion mechanism in the 22nd trial together with the real movement (with a walking sequence of ① ② ③ ④ ③ ② ③ ④ ③) of the user.

As shown in Fig. 4, the performance of the step detection and tight-coupling sensor fusion mechanisms were somewhat similar in terms of accuracy, and in some circumstances one could outperform the other. The overall average estimation error based on the former was 0.47 meters (with a standard deviation of 0.154 meters) and 0.56 meters (with a standard deviation of 0.165 meters) based on the latter.
adopted to reduce power consumption at the expense of losing much detailed acceleration information. However, we believed the results were still promising.

Another issue that also caused notable misestimates of the movement was the compass heading deviation, which could gradually accumulate up to about ±20°, or even worse when the compass was interfered with by a local magnetic field. It is also important to note that the higher accuracy of measuring the orientation of the test environment (i.e. knowing θ in Fig. 1) before telemonitoring, the better the overall accuracy of the step detector. (In our earlier experiments, the step detectors had a maximum error of 1.17 meters and an average error of 0.58 meters.) Although the iPhone would automatically calibrate the compass from time to time, such an occasional drift could cause a deviation of around 0.22 meters in one-step estimation (with step length of about 0.65 metres), or even accumulating to several meters when the user walked multiple steps in a straight line. To calibrate the iPhone’s compass heading estimation, we had tried using iPhone’s gyroscope measurements, including yaw rotation and quaternion, as references for calibration. However, this effort proved to be ineffective, as the gyroscope outputs were even more inaccurate.

Consequently, in our future work, we plan to integrate accurate location information from low-cost force sensitive resistor (FSR) sensors deployed at certain known locations inside the home environment to further improve the accuracy of our localization algorithm. In addition, we considered the estimation results might be very unreliable if the patient makes movements with excessive speed or unusual formations (e.g. walk in strides or lamely) which could cause serious misestimates on step detection. As our experiments were based on a normal subject/user, we also plan to perform further tests and evaluations using a wider range of subjects, including for example Parkinson’s disease patients with a tremor issue, as well as using a variety of working patterns, in the future. Regarding the orientation of the iPhone 5, our intention was to use a small, lightweight sensor tag (e.g. TI SensorTag with on-board inertial sensors) attached to the user’s clothing to minimize the intrusiveness, so during each of the trials the phone was carefully carried by hand to simulate a sensor tag. Nevertheless, by applying a transformation matrix to account for the change in the phone orientation, both the direction of travel and acceleration in a predefined reference frame can still be well retrieved.

### V. Conclusions

In this paper, we have presented a proof-of-concept localization system for real-time indoor patient movement pattern telemonitoring. With both the maximum estimation error well under two meters and an average estimation error of 0.47 meters for step detection (and 0.56 meters for tight coupling sensor fusion mechanism) in our trials, we have achieved the essential part of our research objective.

For future work, one of our priorities is to further adapt our current design to a home-like environment where wireless signal transmission might well be significantly affected by various floor plans, partition walls, and furniture. To this end,
we plan to use more BLE sensors to get better signal coverage across different rooms and lower measurement noise, as well as to deploy FSR sensors to provide accurate location information as mentioned in Section IV. We also plan to develop a mechanism based on the step detection to automatically record where the patient deploys all the BLE and FSR sensors in the home prior to performing real-time location tracking. By doing so, we could enhance not only the reliability and deployability of our system, but also the usability and portability.

REFERENCES


Po-Chou Liang completed his undergraduate education in electronic engineering at National Taipei University of Technology, Taipei, Taiwan, in 1988. He received the master degrees both in computer science from Yuan-Ze University, Taoyuan, Taiwan, in 1994 and in Operational Telecommunications Program from Coventry University, Coventry, UK, in 1998. He is currently pursuing the Ph.D. degree in computing at University of Surrey, Guildford, UK.

From 2006 to present, he serves as Senior Technical Specialist at the National Communications Commission, Taipei, Taiwan. From 2003 to 2006, he doubled as Deputy Convenor of the ICT Development Steering Group, APEC (Asia-Pacific Economic Cooperation) Telecommunications and Information Working Group, and Section Chief at Directorate General of Telecommunications, Taiwan. His research interests include cloud computing use cases in a digital home environment, real-time healthcare telemonitoring, wireless technology, and sensor technology.

In 2011, Mr. Liang was selected by the Taiwan Government as the first of six outstanding government officials to receive full scholarship for conducting Ph.D. studies abroad in several identified areas.

Paul Krause graduated in mathematics and physics from the University of Exeter, Exeter, UK, in 1977. He then completed his PhD in geophysics at the University of Exeter in 1980.

From 1980 to 1987 he worked in low-temperature Physics at the National Physical Laboratory, before moving to the University of Surrey from 1987-89 to work on formal methods in software engineering. He spent the period from 1989-1996 working at the Imperial Cancer Research Fund, working on a range of theoretical issues in Artificial Intelligence. His work specifically worked on automated reasoning under uncertainty, with applications in decision support for medical diagnosis, patient management and toxicological risk prediction. From 1996-2003, he was a Principal Scientist, then Senior Principal Scientist at Philips Research Laboratories, Redhill, UK. Whilst there, he led a number of software engineering research projects supported by global development organizations within Philips. He particularly focused on specification methods, automated testing and software quality assessment. From 2001 he started a joint appointment at the University of Surrey, Guildford, UK, moving to full time Professor of Software Engineering in 2003. His work now focuses on formal modeling of Internet based systems, and the analysis of large-scale imbalanced data sets.

Prof. Krause is a Fellow of the Institute of Mathematics and its Applications.