Evaluation of a Low-Cost MEMS IMU for Indoor Positioning System

Md. Galib Hasan, Md. Kamrul Hasan, Ragib Ahsan, Tania Sultana, R. C. Bhowmik

Abstract— Indoor Positioning Systems (IPS) have gained popularity for their potential application to track people in risky environments or in rescue missions. This paper looks at how an Arduino combined with a foot-mounted inertial measuring unit (IMU) can be used to provide absolute positioning despite the presence of drift in the inertial unit. The IMU we have used contains a tri-axis accelerometer, a tri-axis gyroscope and a tri-axis magnetometer. The orientation was calculated using quaternion output from the IMU which uses gyroscope with drift correction performed by referencing the earth's gravity for pitch and roll and the geomagnetic field from magnetometer for heading. Acceleration signal outputted by the IMU is doubly integrated with time that yields the travelled distance. During the preceding time instants, the position information becomes increasingly untrustworthy and the validity of the position updates decays. So, we used a smoother algorithm based Kalman filter for better estimation accuracy in position estimation. A second method for distance measurement was implemented which uses an algorithm that measures the distance traveled by counting the number of steps. The step length determination was made by an algorithm that takes the angle between legs made by the accelerometer and gyroscope. Experiments were conducted on different scenarios and the results were compared which indicate that the typical positioning accuracy is below 5% for both methods. Issues and proposed improvements to the system are also discussed in this work.

Index Terms— IPS, IMU, Kalman Filter, Arduino, Quaternion.

I. INTRODUCTION

Today, GPS provides localization outdoors, but precise indoor tracking of people remains an open research problem. We have seen indoor location systems based on infra-red, ultrasound, narrowband radio, Wi-Fi signal strength, UWB, vision, and many others [1]. However, few can be easily deployed over large buildings whilst still providing accurate localization. To minimize deployment and infrastructure costs, we wish to develop a wearable location system that can position itself absolutely within a complex structure. One type of sensor which does seem applicable to people tracking is inertial measuring units (IMUs). Recent advances in micro-electro-mechanical systems (MEMS) technologies have made such units smaller and cheaper, however also more prone to error. The main characteristics of the MEMS sensors are: the small dimensions of the capsules, low weight, low power consumption, low cost, small starting time, high performances and a small number of additional components.

Accurate people tracking in a general environment using small and wearable inertial sensors has yet to be reliably shown, although previous attempts have been made [2,3]. Theoretically, single and double integration of the gyro and accelerometer outputs will provide velocity and position information. In practice when working with standard IMU units, the non-linearity and noise present in the sensors make the trajectory prediction valid for short periods of time. So Kalman filter is needed to utilize for solving this problem [4,5].

In this paper we demonstrate how to locate and track a person in a building for which we have an accurate model, using an off-the-shelf wearable inertial system and a Kalman filter to tackle the traditional drift problems associated with inertial tracking. Since the system requires very little fixed infrastructure, the monetary cost is proportional to the number of users, rather than to the coverage area as is the case for traditional indoor location systems. We propose that such a system could be used to enable the deployment of location-aware applications in large buildings, where the installation of a high accuracy absolute location system is either too expensive or impractical.

A. Outline of The Paper

The outline of this paper is as follows. In section 2 and 3 we first review relevant literature and how a foot-mounted IMU might be used to provide a sequence of relative locations and different orientation representation methods. Section 4 covers discussion on two distance measuring algorithms. The experiments, results, and issues are discussed on section 5. At last, the conclusion and proposed further works are presented on section 6.

II. ARDUINO AND IMU

Arduino (Fig-1(a)) is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended not only for engineers or scientists but also for artists, designers, hobbyists and anyone interested in creating interactive objects or environments. In our work we have used Arduino to process data collected from the IMU sensors as well as running the necessary algorithms based on the data received to determine the location.

An inertial measurement unit, or IMU (Fig-1(b)), is an electronic device that measures and reports on a craft's velocity, orientation, and gravitational forces, using a

Manuscript received September 01, 2013.

Md. Galib Hasan, EEE, Pabna University of Science & Technology, Pabna, Bangladesh.
Md. Kamrul Hasan, Embedded System, SinePulse GmbH, Munich, Germany.
Ragib Ahsan, Game Development, Playdom, Dhaka, Bangladesh.
Tania Sultana, Commercial Operation, DESCO, Dhaka, Bangladesh.
R. C. Bhowmik, Mathematics, Pabna University of Science & Technology, Pabna, Bangladesh.

combination of accelerometers and gyroscopes, sometimes also magnetometers. IMUs are typically used to maneuver aircraft, including unmanned aerial vehicles (UAVs), among many others, and spacecraft, including satellites and landers. Recent developments allow for the production of IMU-enabled GPS devices. An IMU allows a GPS to work when GPS-signals are unavailable, such as in tunnels, inside buildings, or when electronic interference is present.

(a) (b)

Fig-1: (a) Arduino Uno (b) 9DOF IMU

A. Arduino

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software running on a computer (e.g. Flash, Processing, MaxMSP). The version of the Arduino we have used is called Arduino Uno and it has the following features:

- ATmega328 microcontroller
- Input voltage - 7-12V
- 14 Digital I/O Pins (6 PWM outputs)
- 6 Analog Inputs
- 32k Flash Memory, 2KB SRAM, 1KB EEPROM
- 16MHz Clock Speed

B. IMU

The IMU we have used is the world's first 9-axis Motion Tracking MEMS device designed for the low power, low cost, and high performance requirements of consumer electronics equipment including smart phones, tablets and wearable sensors. This IMU contains a 3-axis gyroscope, 3-axis accelerometer, and an onboard Digital Motion Processor (DMP) capable of processing complex 9-axis Motion Fusion algorithms; and a 3-axis digital compass. The part's integrated 9-axis Motion Fusion algorithms access all internal sensors to gather a full set of sensor data.

1. Accelerometer

An accelerometer is a device that measures proper acceleration. The proper acceleration measured by an accelerometer is not necessarily the coordinate acceleration (rate of change of velocity). Instead, the accelerometer sees the acceleration associated with the phenomenon of weight experienced by any test mass at rest in the frame of reference of the accelerometer device. For example, an accelerometer at rest on the surface of the earth will measure an acceleration g= 9.81 m/s^2 straight upwards, due to its weight. The triple-axis MEMS accelerometer included in the IMU we have chosen has the following features:

- Digital-output 3-Axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8gand ±16g
- Integrated 16-bit ADCs enable simultaneous sampling of accelerometers while requiring no external multiplexer
- Orientation detection and signaling
- User-programmable interrupts
- High-G interrupt
- User self-test

2. Gyroscope

A gyroscope is a device for measuring or maintaining orientation, based on the principles of angular momentum. Most available MEMS gyroscopes use a tuning fork configuration. Two masses oscillate and move constantly in opposite directions. When angular velocity is applied, the Coriolis force on each mass also acts in opposite directions, which result in capacitance change. This differential value in capacitance is proportional to the angular velocity Ω > and is then converted into output voltage for analog gyroscopes or LSBs for digital gyroscopes. The triple-axis MEMS gyroscope included in the IMU we have chosen has the following features:

- Digital-output X, Y, and Z Axis angular rate sensors (gyroscopes) with a user-programmable full scale range of ±250, ±500, ±1000, and ±2000°/sec
- Integrated 16-bit ADCs enable simultaneous sampling of gyros
- Improved low-frequency noise performance
- Digitally-programmable low-pass filter
- User self-test

2. Magnetometer

A magnetometer is a measuring instrument used to measure the strength and, in some cases, the direction of magnetic fields. There are many approaches for magnetic sensing whereas the most common magnetometer method is the Hall effect method. It works on the principle that a voltage can be detected across a thin metallic element, when the element is placed in a strong magnetic field perpendicular to the element's plane. The triple-axis MEMS magnetometer included in the IMU we have chosen has the following features:

- 3-axis silicon monolithic Hall-effect magnetic sensor with magnetic concentrator
- Wide dynamic measurement range and high resolution with lower current consumption.
- Output data resolution is 13 bit (0.3 µT per LSB)
- Full scale measurement range is ±1200 µT
- Self-test function

III. ORIENTATION REPRESENTATION METHODS

In geometry the orientation, angular position, or attitude of an object such as a line, plane or rigid body is part of the description of how it is placed in the space it is in. Namely, it is the imaginary rotation that is needed to move the object from a reference placement to its current placement. A rotation may not be enough to reach the current placement. It may be necessary to add an imaginary translation, called the object's location (or position, or linear position). The location and orientation together fully describe how the object is placed in space. The above mentioned imaginary rotation and translation may be thought to occur in any order, as the orientation of an object does not change when it translates, and its location does not change when it rotates.

Euler's rotation theorem (Fig-2) shows that in three dimensions any orientation can be reached with a single rotation around a fixed axis. This gives one common way of representing the orientation using an axis-angle representation. Other widely used methods include rotation matrices , Euler angles, or quaternions.

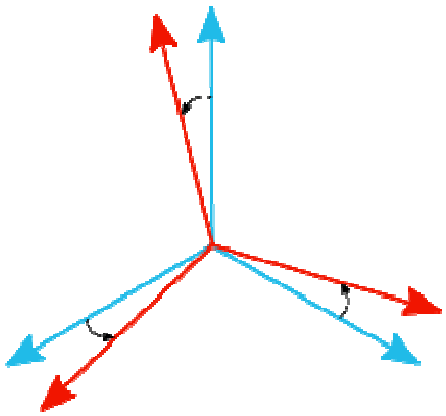


Fig-2: Changing orientation of a rigid body is the same as rotating the axes of a reference frame attached to it.

A. Rotation Matrix

One way to represent the orientation of a coordinate space in 3D is to list the basis vectors of one coordinate space, expressed using the other coordinate space. When these basis vectors are used to form the rows of a 3×3 matrix, we have expressed the orientation in matrix form. Another way to say this is that we can express the relative orientation of two coordinate spaces by giving the rotation matrix that can be used to transform vectors from one coordinate space to the other. The most important property of matrix form is that we can use a matrix to rotate vectors between object and inertial space. No other representation of orientation allows this. However, a matrix uses nine numbers to store an orientation, and it is possible to parameterize orientation using only three numbers. These "extra" numbers can cause some problems. In other words, a matrix contains six degrees of redundancy. There are six constraints that must be satisfied in order for a matrix to be "valid" for representing an orientation. The rows must be unit vectors, and they must be mutually perpendicular.

B. Euler Angle

The Euler angles are three angles introduced by Leonhard Euler to describe the orientation of a rigid body. To describe such an orientation in 3-dimensional Euclidean space three parameters are required. They can be given in several ways, Euler angles being one of them. Euler angles are also used to represent the orientation of a frame of reference relative to another. They are typically denoted as α, β, γ or φ, θ, ψ and named as "heading-pitch-bank" or "yaw-pitch-roll" (Fig-3). The basic idea behind Euler angles is to define an angular displacement as a sequence of three rotations about three mutually perpendicular axes. In this system, an orientation is defined by a heading angle, a pitch angle, and a bank angle. The basic idea is to start with the object in the "identity" orientation — that is, with the axes aligned with the inertial axes. From there, we apply the rotations for heading, then pitch, and finally bank, so that the object arrives in the orientation we are attempting to describe.

Euler angles parameterize orientation using only three numbers, and these numbers are angles. Euler angles are easy for us to use, considerably easier than matrices or quaternions. Perhaps this is because the numbers in an Euler angle triple are angles, which is naturally how people think about orientation. If the conventions most appropriate for the situation are chosen, then the most important angles can be expressed directly.

However, there is one major problem with the Euler angle attitude representation; there exists two attitudes where we have a singularity in the solution (also known as Gimbal lock). Let's have a look at Figure 1 and imagine that the pitch angle is equal to 90 degrees. In this case the yaw and roll perform the same operation. This may not be a problem in computer graphics applications such as in the rendering of the orientation of an object since every attitude does still have a representation, however for applications that require the controls this can be detrimental because we can run into severe math problems when dealing with angles that are close to these singularity points. One way to get around this problem is to add an additional degree so that we have an over defined attitude representation. This is what the quaternion does in a very simplistic sense. The quaternion is based upon the principal of Euler's principal rotation.

C. Quaternion

According to Euler's rotation theorem, any rotation or sequence of rotations of a rigid body or Coordinate system about a fixed point is equivalent to a single rotation by a given angle θ about a fixed axis (called *Euler axis*) that runs through the fixed point. The Euler axis is typically represented by a unit vector $\vec{u}$. Therefore, any rotation in three dimensions can be represented as a combination of a vector $\vec{u}$ and a scalar θ. Quaternions give a simple way to encode this axis–angle representation in four numbers, and to apply the corresponding rotation to a position vector representing a point relative to the origin in $\mathbf{R}^3$.

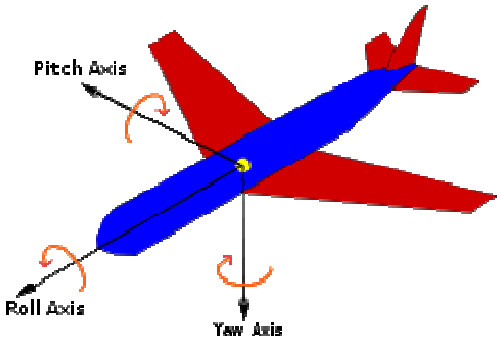


Fig-3: The principal axes of an aircraft

A Euclidean vector such as (2, 3, 4) or (a_x, a_y, a_z) can be rewritten as $2\mathbf{i} + 3\mathbf{j} + 4\mathbf{k}$ or $a_x\mathbf{i} + a_y\mathbf{j} + a_z\mathbf{k}$, where **i**, **j**, **k** are unit vectors representing the three Cartesian axes. A rotation with an angle of rotation of θ around the axis defined by a unit vector

$$\vec{u} = (u_x, u_y, u_z) = u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k}$$

is represented by a quaternion using an extension of Euler's formula:

$$\mathbf{q} = e^{\frac{1}{2}\theta(u_x\mathbf{i}+u_y\mathbf{j}+u_z\mathbf{k})} = \cos\tfrac{1}{2}\theta + (u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k})\sin\tfrac{1}{2}\theta$$

Because of the advantages of quaternion we have used it to determine the orientation of the target object. The IMU gives the direct quaternion output which is accurate enough to determine the heading of a person. We have also used quaternion later to compensate gravity component from raw acceleration in order to get dynamic acceleration while determining distance travelled. But in order to make a live visual representation of the trajectory as we have used Processing and Euler angles were easier to use there, we have simply used the following conversion formulae to convert quaternion in to Euler angles:

$$\begin{bmatrix}\phi\\ \theta\\ \psi\end{bmatrix} = \begin{bmatrix}\arctan\frac{2(q_0q_1+q_2q_3)}{1-2(q_1^2+q_2^2)}\\ \arcsin(2(q_0q_2-q_3q_1))\\ \arctan\frac{2(q_0q_3+q_1q_2)}{1-2(q_2^2+q_3^2)}\end{bmatrix}$$

arctan and arcsin have a result between $-\pi/2$ and $\pi/2$. With three rotations between $-\pi/2$ and $\pi/2$ we can't have all possible orientations. We need to replace the arctan by atan2 to generate all the orientations.

$$\begin{bmatrix}\phi\\ \theta\\ \psi\end{bmatrix} = \begin{bmatrix}\operatorname{atan2}(2(q_0q_1+q_2q_3), 1-2(q_1^2+q_2^2))\\ \arcsin(2(q_0q_2-q_3q_1))\\ \operatorname{atan2}(2(q_0q_3+q_1q_2), 1-2(q_2^2+q_3^2))\end{bmatrix}$$

IV. DISTANCE MEASUREMENT ALGORITHMS

Over the last decades there have been lots of devices that measure traveled distance. The best results have been obtained with GPS, but these systems are useless in closed spaces because a permanent communication with satellites is needed for pinpointing the location and measuring the traveled distance. The main sensors used for determining the distance are: GPS, ultrasonic, infrared, optical, inertial and electromagnetic sensors.

In our work as we have used MEMS sensor i.e. IMU which includes the accelerometer sensor the output of which can be used to calculate distance. If an accelerometer is available, the first solution that one might think of is to integrate twice the acceleration to obtain the total distance. The problem with this approach is that sensors are inexact and their errors would accumulate by integrating. To avoid this, the Kaman filter is used which is basically a statistical method that combines a knowledge of the statistical nature of system errors with a knowledge of system dynamics, as represented by a state space model, to provide an estimate of the state of a system. Alternatively, researchers have come up with methods to detect steps and estimate their lengths without integrating the acceleration.

A. Distance Measurement: Accelerometer & Kalman Filter

An IMU contains three orthogonal rate-gyroscopes and accelerometers, which report angular velocity and acceleration respectively. In principle, it is possible to track the orientation of the IMU by integrating the angular velocity signals. This can then be used to resolve the acceleration samples into the global frame of reference, from which acceleration due to gravity is subtracted. The remaining acceleration can then be integrated twice to track the position of the IMU relative to a known starting point and heading [6]. Unfortunately the error, or 'drift' in the calculated position grows rapidly with time. The main cause of drift is small errors perturbing the gyroscope signals, which cause growing tilt errors in the tracked orientation. A small tilt error causes a component of acceleration due to gravity to be projected onto the globally horizontal axes. This residual is double integrated, causing an error in position which grows cubically in time in the short term. The drift incurred by a high end MEMS IMU will typically exceed 100 meters after 1 minute of operation.

Here comes the Kalman filter to give the solution. The Kalman filter operates by producing a statistically optimal estimate of the system state based upon the measurement(s). To do this it will need to know the noise of the input to the filter called the measurement noise, but also the noise of the system itself called the process noise. To do this the noise has to be Gaussian distributed and have a mean of zero, luckily for us most random noise have this characteristic.

In Fig-4, whole computational process for Kalman filter is laid out in a schematic diagram. Although it bears the name 'filter,' Kalman filter would rather be better considered as a computer algorithm. In the Fig-4, the part in the dashed box is Kalman filter algorithm. The structure is very simple. It receives only one input (measurement, z_k) and returns one output (estimate, $\hat{x}_k^-$). Internal process is done through a four-step computation.

Now let us look into the computational procedure of the algorithm in detail. The first step is for prediction. The two

variables $\hat{x}_k^-$ and P_k^- which will be used throughout the Steps II through IV, are computed in this step. The superscript '-' means predicted value. The formulae in this prediction step have very close relationship with system model. This will be discussed in detail later. In Step II, Kalman gain (K_k) is being computed. The variable P_k^- computed in the previous step is used. H and R are the values preset outside Kalman filter. In Step III, an estimate is computed from a measurement given as input. It has not been clearly revealed yet, but the formula in this step is related to low-pass filter. This will be explained later, as well. The variable $\hat{x}_k^-$ is the one computed in Step I. In Step IV, error covariance is computed. Error covariance is a measure indicating how accurate the estimate is. Normally, decision to trust and use or discard the estimate computed in the previous step is made based on the review of the error covariance.

As each of the parameters is recursively computed based on its previous value, the previous on its previous value till initial condition, the Kalman filter incorporates the information obtained by all the previous values in its prediction. It does so without actually storing that data and uses simple equations in a loop, making it computationally inexpensive. It is also necessary to point out that Kalman gain and error covariance equations are independent of actual observations.

Now we need to make a Kalman filter model for accelerometer. A Kalman filter model for accelerometer is well described in [5]. However, the model chosen was Weiner process acceleration model [7]. This model requires

$$\Phi_k = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}$$

$$Q_k = q \begin{pmatrix} \frac{\Delta t^5}{20} & \frac{\Delta t^4}{8} & \frac{\Delta t^3}{6} \\ \frac{\Delta t^4}{8} & \frac{\Delta t^3}{3} & \frac{\Delta t^2}{2} \\ \frac{\Delta t^3}{6} & \frac{\Delta t^2}{2} & \Delta t \end{pmatrix}$$

And $\Delta t = 0.005$, $q = \sigma_q^2$, $H_k = [001]$, $R_k = [\sigma_r^2]$

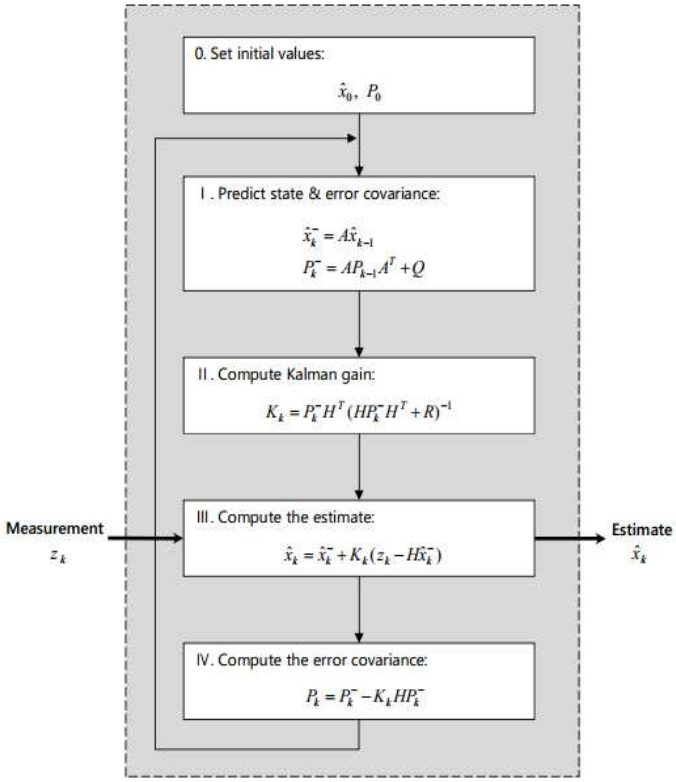


Fig-4: Kalman filter algorithm

Fig-5 shows how the accelerometer bias is modeled as a random walk process. The bias is modeled as an integrated white noise with a power spectral density (PSD) being W. The complete process is modeled by three integrators in cascade [8, p. 232]. From this model, the exact expressions for Φ_k and Q_k can be worked out to the form as shown above. The power spectral density of the input white noise W is 1 $(m/s^2)^2/(rad/s)$ and the sampling time Δt equals 1/206.6s.The value of W was obtained after performing some experiments aimed to provide better results.

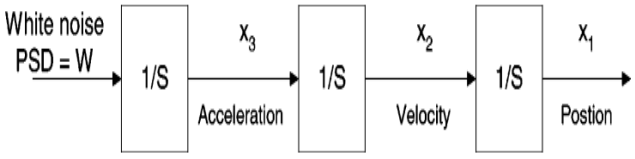


Fig-5: The process model of the accelerometer data for the Kalman filter

In order to remove the effect of gravity error, the raw acceleration values were gravity-compensated to get the dynamic acceleration before applying the Kalman filter algorithm. This was simply done using the quaternion as follows:

```
dyn_accel_x =  raw_accel_x - 2(q1*q3 - q0*q2)
dyn_accel_y = raw_accel_y - 2(q0*q1 + q2*q3)
dyn_accel_z =
raw_accel_z - (q0*q0 - q1*q1 - q2*q2 + q3*q3)
```

The idea behind this is pretty simple. With the quaternion we can compute the expected direction of gravity and then

subtract that from the accelerometer readings. The result is the dynamic acceleration.

B. Distance Measurement: Step Detection

There are several ways to do step detection using accelerometer information, but the main difference strives on whether the sensor is located on the foot, waist, or in a different place. Since this project deals with positioning using an IMU, the sensors are assumed to be on the feet.

1. Counting The Number of Steps

We have used the motion interrupt feature of the IMU to detect and count the number of steps. The idea was to set a threshold value of acceleration within a certain motion period to detect the step. The threshold values were determined after some trials and errors.

The graph in Fig-6 shows the total acceleration measured by the accelerometer while walking. After several tests it was observed that every step was introducing a significant increase in the acceleration during a short instant. These acceleration increments are in the form of sharp peaks in the graph. An approach to identify peaks in the acceleration was taken. A peak is detected when the acceleration tend changes from incrementing to decrementing.

The arrows in the graph identify peaks. However a step normally introduces a peak that reaches at least 13m/s^2, so a threshold of 12.5m/s^2 was introduced to discard small peaks and also be sure not to miss any step. Another phenomenon which was observed is that sometimes one step could generate more than one peak. This issue was solved adding the requisite that the previous step should have happened at least 350 ms before. The same flow control of the total acceleration is used to detect motion. When during a period of at least 450 ms (25 samples) no peaks are detected, then the current state is set to no motion. As soon as there is a peak the state is immediately set back to motion.

The two described procedures result into an acceptable performance. However possible improvements can be done, especially in the detection of false positives (steps or motion detected when the user is not moving but shakes the device).

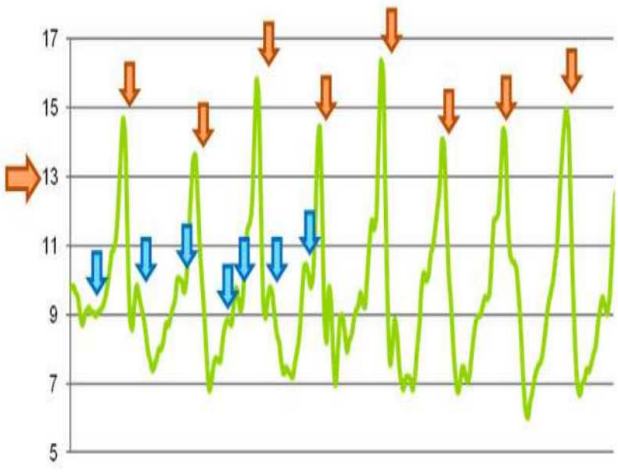


Fig-6: Step and motion detection

2. Determination of Step Length

For the measuring of the distance we have resorted to the counting of the steps. And for measuring the distance of every step we determined the angle between the legs (Fig-7). The device was attached in the upper part of the knee. If we know the length of a leg and the angle between the legs we can easily calculate the distance walk using the law of cosines.

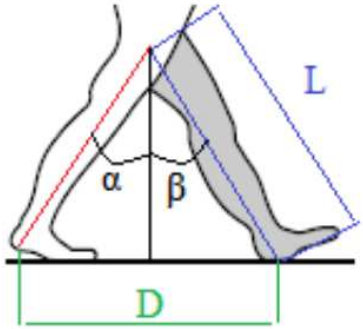


Fig.-7: Legs Angles

Where:
L – The length of the leg
D – Step distance
α, β – Angles between legs
Appling the low of cosines we have:

$$D^2 = L^2 + L^2 - 2L^2 cos\,(\alpha + \beta)$$

$$D = L \times \sqrt{2 \times \left[1 - \cos(\alpha + \beta)\right]}$$

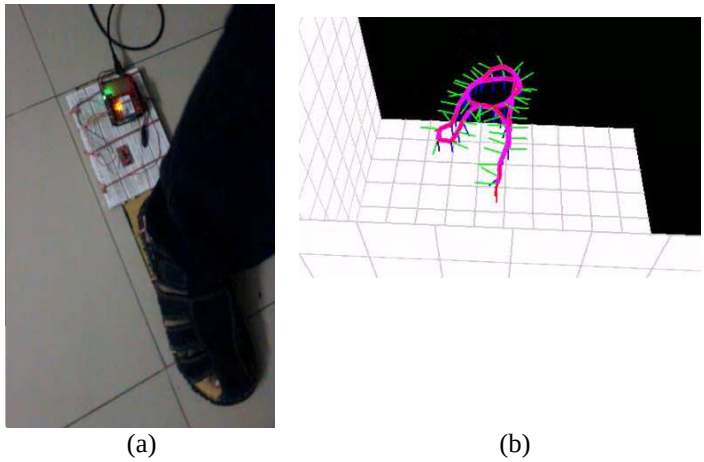
And for obtaining the final distance we have used the following equation:

$$D_{total} = \sum_{n=1}^{N} L \times \sqrt{2 \times \left[1 - \cos(\alpha + \beta)\right]}$$

where N represents the number of steps.

V. EXPERIMENTAL RESULTS

In this experiment a user has walked through a typical indoor environment (Fig-8(a)). At first it was experimented to check how accurate the orientation i.e. heading was. In order to do



(a) (b)

Fig-8: (a) User with foot-mounted IMU (b) Recorded sample trajectory of user movement at indoor

so the user has started walking from one room and walked straight for a while and then made a circular shaped loop to test the heading angle accuracy. Then he entered into the next room and finally returned to his starting position through the shortest possible path making a smooth turn. The whole trajectory was recorded (Fig-8(b)) and later matched to the original one. The result was satisfactory as the accuracy was good enough for human positioning. Later, for the distance measurement, the device was tested on a 50 meter range. The first 5 measurements were made in normal walking conditions without stopping. The next five measurements were made with a low walking speed, and next five were done at high speed. The last three were made with stops. The results are represented in the Table-1.

Table -1: Comparison of real and measured distance

No. Of Try	Real Steps	Measured Steps	Real Distance (m)	Measured Distance (m)	
				Step Detect	Kalman
01	85	87	50	46	47
02	85	86		47	48
03	84	85		46	48
04	85	84		48	49
05	86	84		47	48
06	94	97		49	50
07	95	96		51	51
08	94	94		52	53
09	93	94		51	52
10	95	94		49	51
11	80	80		46	48
12	80	81		45	49
13	79	78		47	47
14	81	78		45	46
15	81	82		46	48
16	85	85		48	49
17	85	86		49	51
18	86	84		48	49

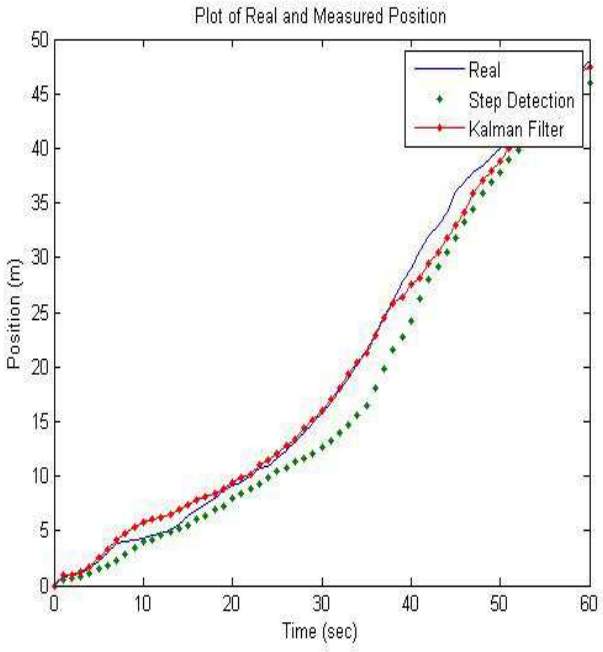


Fig-9:Comparison of step detection & Kalman filter method

The Fig-9 shows the real distance against measured distance curve for both methods. It is seen that the performance of the Kalman filter method was slightly better than the step detection method. The overall results for the positioning system show an average error rate in position of less than 5% with only one test over this result. It is important to note that only one solution was used for all the experiments, meaning that no changes were made to the Kalman filter. The probable reason of errors could be the movement of the device during experiment and the thermal bias drift of the accelerometer as well as magnetic disturbance.

VI. CONCLUSIONS

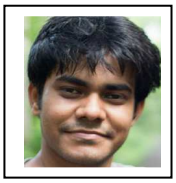
The experimental results have provided a useful evaluation of a low-cost solid state IMU. The performance of the accelerometer is shown to be acceptable as a short-duration distance-measuring device for mobile platform or robot . So, we applied a smoothing Kalman filter algorithm in the absence of absolute positioning information that constructed the best estimate of the position over a time period using all the measurements in that time interval. Further research would focus on the better model for stochastic bias of solid state accelerometer in order to reduce the effect of bias and thermal noise and fuse the solid state accelerometer data with other positioning technologies like Wi-Fi or FM. Also, considering the map information and an adaptive step length algorithm could help to improve the overall performance of the system.

REFERENCES

[1] M. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggles, A. Ward, and A. Hopper. Implementing a sentient computing system. Computer, 34(8):50–56, 2001.
[2] Foxlin. Pedestrian tracking with shoe-mounted inertial sensors. Computer Graphics and Applications, IEEE, 25(6):38–46, 2005.
[3] S. Godha, G. Lachapelle, and M. E. Cannon. Integrated GPS/INS system for pedestrian navigation in a signal degraded environment. In ION GNSS 2006, 2006.
[4] Lawrence A.,"Modern Inertial technology", 1993, springer verlag.
[5] R.G. Brown and P.Y.C. Hwang, "Introduction to Random signals and Applied Kalman filtering".
[6] D. Titterton and J. Weston. Strapdown Inertial Navigation Technology. The American Institute of Aeronautics and Astronautics, 2nd edition, 2004.
[7] Y. Bar-Shalom, X. Rong Li, T. Kirubajan, Estimation with applications to tracking and navigation, Wiley-Interscience, 2001.
[8] Gil, J.-S., Cho, Y.-D., and Kim, S.-H.: Design of a low-cost inertial navigation system with GPS for car navigation system, in:Proc. of the 5th World Congress on ITS, 1998.
[9] Chin-Woo Tan and Sungsu Park, GPS-Aided gyroscope-free inertial navigation systems, WebPage of PATH, University of California at Berkeley.
[10] InvenSense, MPU-9150 Datasheet.
[11] Hugh H. S. Liu and Grantham K. H. Pank Member IEEE, "Accelerometer for mobile robot positioning,"IEEE Transactions on Industry Applications, Vol.37, No.3, May/June 2001.
[12] Myungsoo Jun, Stergios I. Roumeliotis, Graurav S. Sukhatme, "State estimation of an autonomous helicopter using kalman filtering,"Proceeding of the 1999 IEEE/RSJ, International Conference on Intelligent Robots and Systems.

Md. Galib Hasan has been graduated from Khulna University of Engineering & Technology in 2009. His major is Electrical & Electronic Engineering and currently working as a lecturer in the dept. of EEE at PUST. His research interest includes embedded systems, VLSI, Chip Design etc.

Md. Kamrul Hasan is a graduate of Computer Science & Engineering. He has been working as an embedded system engineer since 2009. Currently he is jointly working with SinePulse GmbH and Infineon Technologies at Munich, Germany.



Ragib Ahsan is a Computer Science & Engineering Graduate. He has been working as a software engineer at Playdom Bangladesh since 2012. His current skill is social game development as well as IOS application development.

Tania Sultana has completed her graduation in Electrical and Electronic Engineering in 2009. Then she has worked at AREVA T & D for about 1.5 years. Currently she is working as an assistant engineer at Dhaka Electric Supply Company (DESCO).

Rajendra Chandra Bhowmik has completed M.Sc. in Mathematics from Chittagong University. Currently he is working as Asst. Professor in the department of Mathematics at PUST. His research interest includes Group theory, Number theory, Topology etc.