Android accelerometer accuracy (Inertial navigation)

Asked 13 years, 2 months ago Modified 4 months ago

Viewed 108k times Part of Mobile Development Collective

I was looking into implementing an Inertial Navigation System for an Android phone, which I realise is hard given the accelerometer accuracy, and constant fluctuation of readings.

To start with, I set the phone on a flat surface and sampled 1000 accelerometer readings in the X and Y directions (parallel to the table, so no gravity acting in these directions). I then averaged these readings and used this value to calibrate the phone (subtracting this value from each subsequent reading).

I then tested the system by again placing it on the table and sampling 5000 accelerometer readings in the X and Y directions. I would expect, given the calibration, that these accelerations should add up to 0 (roughly) in each direction. However, this is not the case, and the total acceleration over 5000 iterations is nowhere near 0 (averaging around 10 on each axis).

I realise without seeing my code this might be difficult to answer but in a more general sense...

Is this simply an example of how inaccurate the accelerometer readings are on a mobile phone (HTC Desire S), or is it more likely that I've made some errors in my coding?

MD

android physics accelerometer acceleration calibration

Share

Improve this question

Follow

edited Apr 30, 2016 at 10:47

Ciro Santilli OurBigBook.com

380k ● 117 ● 1.3k ● 1.1k

asked Oct 19, 2011 at 22:40

woodstock365

1,790 ● 4 ● 19 ● 35

3 webvr-polyfill is a great source of inspiration: github.com/borismus/webvr-polyfill/tree/master/src look at how they polyfill a VR sensor using accelerometer data: github.com/borismus/webvr-polyfill/blob/master/src/… – S.C. Mar 13, 2016 at 11:42

Question that also considers gyroscope: stackoverflow.com/questions/8264518/… – Ciro Santilli OurBigBook.com Apr 30, 2016 at 8:14

6 Answers

Sorted by: Highest score (default)

You get position by integrating the linear acceleration twice but **the error is horrible. It is useless in practice.**

Here is an explanation why (Google Tech Talk) at 23:20. I highly recommend this video.

It is not the accelerometer noise that causes the problem but the gyro white noise, see subsection 6.2.3 Propagation of Errors. (By the way, you will need the gyroscopes too.)

As for indoor positioning, I have found these useful:

RSSI-Based Indoor Localization and Tracking Using Sigma-Point Kalman Smoothers

Pedestrian Tracking with Shoe-Mounted Inertial Sensors

Enhancing the Performance of Pedometers Using a Single Accelerometer

I have no idea how these methods would perform in real-life applications or how to turn them into a nice Android app.

A similar question is this.

UPDATE:

Apparently there is a newer version than the above Oliver J. Woodman, "An introduction to inertial navigation", his PhD thesis:

Pedestrian Localisation for Indoor Environments

Share Improve this answer

Follow

edited May 23, 2017 at 12:18

Community Bot

1 ● 1

answered Oct 20, 2011 at 12:35

Ali

58.3k ● 31 ● 173 ● 272

2 I realise this is a long time ago, but I've got a follow-up question. The camera in Android JB has a 'panorama' feature, which lets you take a panoramic picture by moving the phone, either rotating it *or* moving it linearly along one axis. To do this it has to track the position of the phone relatively accurately - at least better than the 20cm/s error mentioned in the video this answer links. How does it do it? Does it have some way of improving the quality of inertial tracking? Or does it use clever image processing to do it using only the camera? – Tom Jun 14, 2013 at 10:56

1 @Tom I believe the latter, the phone concatenates together the pictures purely by image processing algorithms. What makes you think that the phone has to track its position to produce a panorama picture? It was possible to do it with ordinary cameras back in the 90's and clearly, we did not have accelerometers in the cameras back then :) Of course, the pictures were concatenated on an ordinary PCs. But you don't need the position for this, image processing algorithms are sufficient. Hope this helps. – Ali Jun 14, 2013 at 11:09

1 It's quite different to the old manually-take-some-pictures-then-stitch-them-later job. It does track its position in real time somehow. It's a little difficult to explain without demonstrating it. You don't have to take pictures manually - the phone decides when you've moved far enough to take another one. While you're taking the pictures, it shows you a little bar at the bottom with a preview of the panorama. If you point the camera too far down (for instance) it starts beeping and

showing an up arrow to tell you you need to move it back up. – Tom Jun 14, 2013 at 11:39

4 Actually it does seem to use image processing - starting a panorama and then waving your hand in front of the camera will confuse its position tracking system quite badly! – Tom Jun 14, 2013 at 11:40

@Tom OK. I think it mainly uses image processing (as your last comment suggests it as well) but it is likely to be combined with tracking the *orientation* (but not position). – Ali Jun 14, 2013 at 12:53

20

I am just thinking out loud, and I haven't played with an android accelerometer API yet, so bear with me.

First of all, traditionally, to get navigation from accelerometers you would need a 6-axis accelerometer. You need accelerations in X, Y, and Z, but also rotations Xr, Yr, and Zr. Without the rotation data, you don't have enough data to establish a vector unless you assume the device never changes it's attitude, which would be pretty limiting. No one reads the TOS anyway.

Oh, and you know that INS drifts with the rotation of the earth, right? So there's that too. One hour later and you're mysteriously climbing on a 15° slope into space. That's assuming you had an INS capable of maintaining location that long, which a phone can't do yet.

A better way to utilize accelerometers -even with a 3-axis accelerometer- for navigation would be to tie into GPS to calibrate the INS whenever possible. Where GPS falls

short, INS compliments nicely. GPS can suddenly shoot you off 3 blocks away because you got too close to a tree. INS isn't great, but at least it knows you weren't hit by a meteor.

What you could do is log the phones accelerometer data, and a lot of it. Like weeks worth. Compare it with good (I mean really good) GPS data and use datamining to establish correlation of trends between accelerometer data and known GPS data. (Pro tip: You'll want to check the GPS almanac for days with good geometry and a lot of satellites. Some days you may only have 4 satellites and that's not enough) What you might be able to do is find that when a person is walking with their phone in their pocket, the accelerometer data logs a very specific pattern. Based on the datamining, you establish a profile for that device, with that user, and what sort of velocity that pattern represents when it had GPS data to go along with it. You should be able to detect turns, climbing stairs, sitting down (calibration to 0 velocity time!) and various other tasks. How the phone is being held would need to be treated as separate data inputs entirely. I smell a neural network being used to do the data mining. Something blind to what the inputs mean, in other words. The algorithm would only look for trends in the patterns, and not really paying attention to the actual measurements of the INS. All it would know is `historically, when this pattern occurs, the device is traveling and 2.72 m/s X, 0.17m/s Y, 0.01m/s Z, so the device must be doing that now.` And it would move the piece forward accordingly. It's important that it's

completely blind, because just putting a phone in your pocket might be oriented in one of 4 different orientations, and 8 if you switch pockets. And there's many ways to hold your phone, as well. We're talking a lot of data here.

You'll obviously still have a lot of drift, but I think you'd have better luck this way because the device will know when you stopped walking, and the positional drift will not be a perpetuating. It knows that you're standing still based on historical data. Traditional INS systems don't have this feature. The drift perpetuates to all future measurements and compounds exponentially. Ungodly accuracy, or having a secondary navigation to check with at regular intervals, is absolutely vital with traditional INS.

Each device, and each person would have to have their own profile. It's a lot of data and a lot of calculations. Everyone walks different speeds, with different steps, and puts their phones in different pockets, etc. Surely to implement this in the real world would require number-crunching to be handled server-side.

If you did use GPS for the initial baseline, part of the problem there is GPS tends to have it's own migrations over time, but they are non-perpetuating errors. Sit a receiver in one location and log the data. If there's no WAAS corrections, you can easily get location fixes drifting in random directions 100 feet around you. With WAAS, maybe down to 6 feet. You might actually have better luck with a sub-meter RTK system on a backpack to at least get the ANN's algorithm down.

You will still have angular drift with the INS using my method. This is a problem. But, if you went so far to build an ANN to pour over weeks worth of GPS and INS data among n users, and actually got it working to this point, you obviously don't mind big data so far. Keep going down that path and use more data to help resolve the angular drift: People are creatures of habit. We pretty much do the same things like walk on sidewalks, through doors, up stairs, and don't do crazy things like walk across freeways, through walls, or off balconies.

So let's say you are taking a page from Big Brother and start storing data on where people are going. You can start mapping where people would be expected to walk. It's a pretty sure bet that if the user starts walking up stairs, she's at the same base of stairs that the person before her walked up. After 1000 iterations and some least-squares adjustments, your database pretty much knows where those stairs are with great accuracy. Now you can correct angular drift and location as the person starts walking. When she hits those stairs, or turns down that hall, or travels down a sidewalk, any drift can be corrected. Your database would contain sectors that are weighted by the likelihood that a person would walk there, or that this user has walked there in the past. Spatial databases are optimized for this using `divide and conquer` to only allocate sectors that are meaningful. It would be sort of like those MIT projects where the laser-equipped robot starts off with a black image, and paints the maze in memory by taking every turn, illuminating where all the walls are.

Areas of high traffic would get higher weights, and areas where no one has ever been get 0 weight. Higher traffic areas are have higher resolution. You would essentially end up with a map of everywhere anyone has been and use it as a prediction model.

I wouldn't be surprised if you could determine what seat a person took in a theater using this method. Given enough users going to the theater, and enough resolution, you would have data mapping each row of the theater, and how wide each row is. The more people visit a location, the higher fidelity with which you could predict that that person is located.

Also, I highly recommend you get a (free) subscription to GPS World magazine if you're interested in the current research into this sort of stuff. Every month I geek out with it.

Share Improve this answer

Follow

edited May 1, 2015 at 19:15

AppleGrew

9,550 ● 25 ● 81 ● 125

answered Dec 28, 2013 at 9:09

RyanJMcGowan

1,515 ● 1 ● 16 ● 33

"would be to tie into GPS to calibrate the INS whenever possible. Where GPS falls short, INS compliments nicely. " This is what Kalman filtering is for, as I understand it. It combines the strengths of each method to cancel out the weaknesses of the other – endolith Sep 23, 2016 at 15:39

9

I'm not sure how great your offset is, because you forgot to include units. ("Around 10 on each axis" doesn't say much. :P) That said, it's still likely due to inaccuracy in the hardware.

The accelerometer is fine for things like determining the phone's orientation relative to gravity, or detecting gestures (shaking or bumping the phone, etc.)

However, trying to do dead reckoning using the accelerometer is going to subject you to a lot of compound error. The accelerometer would need to be insanely accurate otherwise, and this isn't a common use case, so I doubt hardware manufacturers are optimizing for it.

Share Improve this answer

Follow

answered Oct 19, 2011 at 23:22

Trevor Johns

15.8k ● 3 ● 57 ● 54

> Thanks for the answer. The accelerometers read around -0.8 ms^-2 on both X and Y axes when stationary, so I used this as my offset. By the "Around 10" bit, I meant that over 5000 iterations, adding up each of the accelerations on a single axis from the sensor did not total roughly 0 ms^-2 (as it would if it fluctuated evenly above and below the offset value), but instead tended to register acceleration more in one direction, which after double integration to find position, worked out as the phone moving around 3m in a minute. – woodstock365 Oct 20, 2011 at 18:21

+1 for the use of the aviation navigational term, "dead reckoning." Although dead reckoning would more aptly apply to navigating with a camera than an INS. – RyanJMcGowan Dec 28, 2013 at 7:14

8

Android accelerometer is digital, it samples acceleration using the same number of "buckets", lets say there are 256 buckets and the accelerometer is capable of sensing from -2g to +2g. This means that your output would be quantized in terms of these "buckets" and would be jumping around some set of values.

To calibrate an android accelerometer, you need to sample a lot more than 1000 points and find the "mode" around which the accelerometer is fluctuating. Then find the number of digital points by how much the output fluctuates and use that for your filtering.

I recommend Kalman filtering once you get the mode and +/- fluctuation.

Share Improve this answer

Follow

answered May 31, 2012 at 12:31

Alex Stone

47.3k ● 59 ● 237 ● 418

1 I was looking for calibrating methods. Seems your suggestion is what I need. I just need to confirm. Once I find the mode, say it is 0.5. I didn't get the "Then find the number of digital points by how much the output fluctuates and use that for your filtering." Could you please elaborate on it more. – Nazerke Mar 26, 2013 at 19:48

1 Let's say your accelerometer has 256 output points and fluctuates by 0.015m/s^2 between readings. When you lay your device on the table, your output may be fluctuating in even multiples of 0.015m/s^2. Let's say you get a reading of 0 +/- (X * 0.015). You need to find X (which would be an even number). For example my X may be 3. In this case, I would ignore changes in accelerometer reading that are less than 0.045 m/s^2 – Alex Stone Apr 9, 2013 at 15:03

so android phones accelerometers aren't that good yet..correct? – Techsin Nov 13, 2013 at 4:57

7

I realise this is quite old, but the issue at hand is not addressed in ANY of the answers given.

What you are seeing is the linear acceleration of the device including the effect of gravity. If you lay the phone on a flat surface the sensor will report the acceleration due to gravity which is approximately `9.80665 m/s2`, hence giving the 10 you are seeing. The sensors are inaccurate, but they are not THAT inaccurate! See here for some useful links and information about the sensor you may be after.

Share Improve this answer

Follow

edited May 23, 2017 at 11:54

Community Bot

1 ● 1

answered Mar 3, 2013 at 20:49

Simon O'Hanlon

59.9k ● 15 ● 144 ● 188

19 No - I think you've misread the question: "...readings in the X and Y directions (parallel to the table, so no gravity acting in these directions)". The 9.8 /s2 would be on the Z axis.
– teapot7 May 20, 2013 at 23:45

1

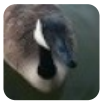
You are making the assumption that the accelerometer readings in the X and Y directions, which in this case is entirely hardware noise, would form a normal distribution around your average. Apparently that is not the case.

One thing you can try is to plot these values on a graph and see whether any pattern emerges. If not then the noise is statistically random and cannot be calibrated against--at least for your particular phone hardware.

Share Improve this answer

Follow

answered Apr 17, 2018 at 4:14



Sarsaparilla

6,650 ● 1 ● 34 ● 22