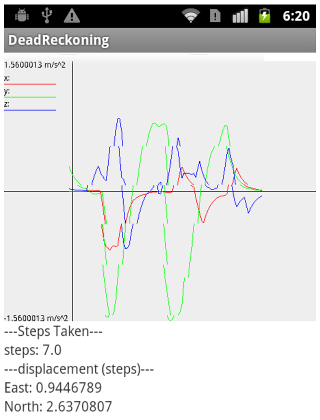
ECE155: Engineering Design with Embedded Systems Winter 2013

Lab 3 (Dead Reckoning) — Week 7

Prepared by Kirill Morozov *version 1*



```
---Steps Taken---
steps: 7.0
---displacement (steps)---
East: 0.9446789
North: 2.6370807
```

Figure 1: A complete lab 3 implementation.

1 Objectives

The goal of this lab is to integrate readings from multiple sensors to make more complex conclusions. In particular, you will combine your solution from lab 2 with readings from your Android phone's compass to track which direction the user is walking in. The solution for Lab 3 will tell the user the distance travelled from the starting point.

During this lab, you will:

1. Filter raw sensor data to account for noise and bias.
2. Study raw rotation readings to identify patterns.
3. Read information about the phone's heading.
4. Combine sensor readings to track a user's displacement.

2 Background

This lab builds on what you learned in Labs 1 and 2. Most students will probably start with their Lab 2 code as a template.

The following information may be helpful in completing this lab:

Compass error sources. For this lab, you will be using the `SensorManager.getOrientation()` method to figure out how the user is oriented relative to the physical world. There are a number of pitfalls to avoid here. First, calculating the orientation of the phone relies on the phone's magnetic field sensor to find magnetic north. So, when you're close to magnetic objects (your computer's speakers, the big green power boxes in the Davis Centre or even the re-bar in the concrete walls), your orientation sensor's readings will be off. We will take this into account when running the lab demos, but you should keep this in mind when testing your application.

Compass calibration. The magnetic field sensors (and hence the compass readings) seem to get stuck sometimes such that they consistently return bogus values. Rotating the phone 1–2 times along each of its axes often fixes this. Alternately, there's also mention of drawing some sort of figure-8 in the air with the phone. People on the Internet call this "calibration".

Compass noise. You will notice that, like the accelerometer, the orientation sensor also has noise. However, you can't directly filter or average the results of your `getOrientation()` calls. This sensor differs from the accelerometer from Lab 2 because the values for the azimuth (the rotation around the X axis) are reported modulo 2π. For example, if the phone reports that its azimuth is 2π at time t and 0 at time $t + 1$, then averaging those values together will give you the completely wrong answer.

A particular danger of taking your bearing at the same point of every step is systemic bias. If you choose to make a widget to draw a compass on the screen (not required), you may notice that it consistently jumps wildly at a particular point of each step. Hence, taking the compass bearing at the same point in each step carries the danger of sending you way off course.

One way to solve this problem is to find a way to average data independent of the data's actual value—you'll want to use only its relative value. For example, at time $t = n$ you can take the value of your variable, and call that the base-line. Then, for some number of samples, you take the average of the (signed) distance between the samples and the base-line.

Another potential solution is to just take periodic sensor readings and trust (or ensure, e.g. via randomness) that these aren't in resonance with the tester's step frequency.

Determining direction. We can think of two ways to determine which direction the user is walking in. You can either assume that the user is always moving in the direction that the phone is pointed at (its heading), or you can look at the direction the phone is accelerating. Using the output acceleration is tricky because of the rotation problem, as discussed below. Just using the phone's heading is sufficient for this lab and for lab 4.

3 The Rotation Problem

In lab 2, you probably noticed that if you rotate the phone on its axis, you get spikes of acceleration on one axis of the phone, but no corresponding counter-spike when the phone stops. We understand that this is a limitation of the accelerometer hardware, which is not set up to detect angular acceleration. In principle, one should be able to detect rotation using the rotation sensor, and account for rotation-based acceleration that way. However, our experience is that the rotation sensor data on the Nexus One phones is too noisy. We believe that this is because the Nexus One does not have an actual gyroscope—its rotation sensor is implemented in software by integrating data from the compass and accelerometer.

However, it should be possible to solve the rotation problem if you have a phone with a gyroscope. If you can find a way to mitigate errors in the rotation data, either by using a gyroscope, or by doing something clever with the accelerometer data, you will get a bonus on this lab. You will need to output your filtered linear acceleration into a graph view during your demo to show that you are correctly filtering your data.

4 Phases of the lab

This lab contains the following steps:

1. Create an Android application project in Eclipse.
2. Gather rotation data for different types of walking. How does the phone's reported orientation correlate with which direction you are moving in?
3. Convert rotation readings into a heading (direction).
4. Correlate your heading with your pedometer to calculate distance travelled.
5. Integrate and test.
6. Demonstrate.

5 Alternate Options

In the past, students have asked for more open-ended labs. We're willing to work with you to make things more interesting, and there is substantial room for creativity in this lab. If you have some other idea your group would like to do for lab 3, talk to Patrick Lam; together, we will develop the idea and an evaluation scheme for it. As a general rule, Lab 3 should involve combining the inputs from two or more sensors and should involve the user performing some sort of action. Some possible ideas are:

Proximity Sensor Some phones have proximity sensors. You could make an application that would detect the presence of objects as the phone was rotated in a sphere and construct a map of the environment.

Gesture Detection You could use the accelerometer to detect how the user moves the phone and report when they have made a particular gesture, like drawing a letter in the air. (This example uses fewer sensors, but requires more complex testing and analysis.)

6 Important Reminders

1. Remember to name your project **Lab3_SSS_XX** where SSS is your lab section (e.g. 201) and XX is your team number (e.g. 02). Unique naming helps us manage the project submissions. There may also be problems if you attempt to load two applications with the same package name on the same device.
2. Different hardware has different levels of noise and bias. Your application will probably work fine if tested on a different phone than it was implemented on. But, beware!

7 Demonstration

Same as with all the previous labs. Don't plagiarise.

Requirements. Your app must:

1. display the current displacement from an initial point on two orthogonal axes (probably North/South and East/West);
2. provide a button which allows users to zero the current displacement ("clear").

We will measure your direction-enhanced pedometer on three walks, each between 50 and 100 steps. We'll throw out the worst result. For each walk, the teaching assistant will walk out of the lab, walk to the other door, and then return to the starting point. We will pre-calculate the displacement (in steps) of a reference point on the walk. For full marks, your application should match our pre-calculated value for the reference point and should read a displacement of zero on both axes when the teaching assistant completes the walk. It is ok to have an error on each axis that is up to 10% of the total walk distance (number of steps). Again, you may choose how the teaching assistant holds the phone during the test.

8 Submission of project files

Upload the version of your code used in the demo to your subversion repository and commit it. The address is: `https://ecesvn.uwaterloo.ca/courses/ece155/w13/groups/group-NNN-MM`

Email Patrick Lam if you can't commit to that address.