Christina Dao

Real-Time Location Tracking

GPS provides accurate outdoor location but consumes a good amount of energy, so having an application that tracks real-time location with *just* the GPS would not be energy efficient. A solution to this problem that I have implemented is dead reckoning achieved by combining GPS data with accelerometer data.

The accelerometer is a low-power sensor that is built into our Android devices, however, these devices accumulate drift, which essentially means that over time, the measurements will become less and less accurate.

I have decided to collect the GPS location of the device every 60 seconds and update the map and marker. Between the 60 second intervals, the accelerometer will constantly collect data. Accelerometer returns a measurement in meter/s/s units; so I keep track of the change in time so that I may determine the distance traveled based on the accelerometer measurement.

@Override

public void onLocationChanged(@NonNull Location location) {

// Every 60 seconds the location and, essentially, calibration is reset

// By updating the marker using the GPS

lat = location.getLatitude();

longt = location.getLongitude();

// Create Bundle

Bundle loc = new Bundle();

loc.putDouble("lat", lat);

loc.putDouble("longt", longt);

getParentFragmentManager().setFragmentResult("coordinates", loc);

}

@Override

public void onSensorChanged(SensorEvent event) {

// Marker is constantly updated by adding to the previous GPS location

// using calculations acquired from accelerometer data.

if (event.sensor.getType() == Sensor.*TYPE\_ACCELEROMETER*) {

// NOTE: Device is always laying flat, faced up for this application, it never rotates either

acclx = (double) event.values[0];

accly = (double) event.values[1];

// Change in time acquire to determine distance travelled given accelerometer data

if (gTimestamp <= 0) {

gTimestamp = event.timestamp;

return;

}

long dT = event.timestamp - gTimestamp;

gTimestamp = event.timestamp;

dT \*= 0.000000001f; // Convert NanoS to S

// Get distance moved in North/South and East/West directions in meters

xdelta = acclx \* (dT \* dT);

ydelta = accly \* (dT \* dT);

// Send data to MapFragment

Bundle XYdelta = new Bundle();

XYdelta.putDouble("xdelta", xdelta);

XYdelta.putDouble("ydelta", ydelta);

getParentFragmentManager().setFragmentResult("distance", XYdelta);

}

}

After calculating the distance traveled in the East to West (the device’s x axis) and the North and South (the device’s y axis) directions, I convert that distance to the longitudinal and latitudinal degree equivalent, and add it to the GPS location that was previously measured.

double lat = (double) (result.getDouble("ydelta") / latConv) + marker.latitude;

double longt = (double) (result.getDouble("xdelta") / (longtConv \* Math.*cos*(Math.*toRadians*(lat)))) + marker.longitude;

Essentially, I am constantly running a double integration on accelerometer measurements to get the distance traveled, converting to to longitudinal and latitudinal degrees, and summing this to the GPS location which is measured every minute. By doing so, we can provide a relatively accurate and constant real-time location tracking. To reiterate, the GPS measurement is taken every minute to eradicate drift of the accelerometer, and essentially recalibrate and maintain accuracy of the location tracker.

I use two fragments in my application, where one acquires measurements from the GPS and accelerometer, and passes it to the second fragment, which is the MapFragment, to display the information on the google map.

DISCLAIMER: This application assumes that the device pose is always laying flat on its back, screen facing up, and never turns.