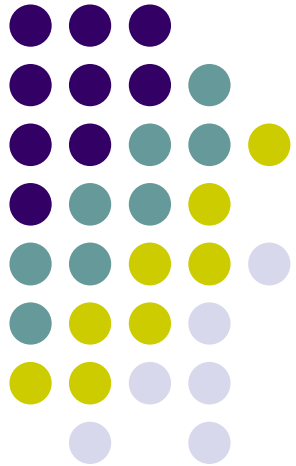
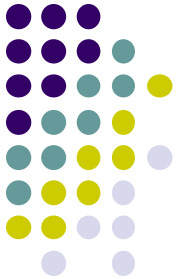
CS 528 Mobile and Ubiquitous Computing

Lecture 6b: Step Counting & Activity Recognition

Emmanuel Agu

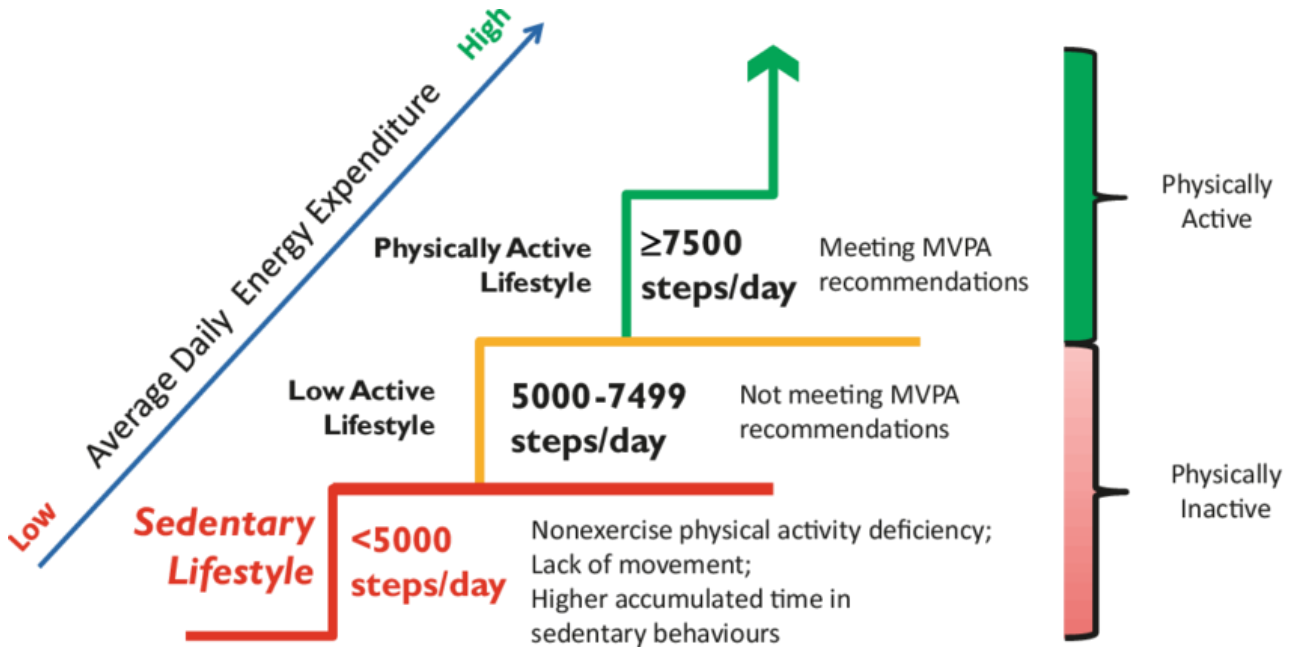




Step Counting
(How Step Counting Works)

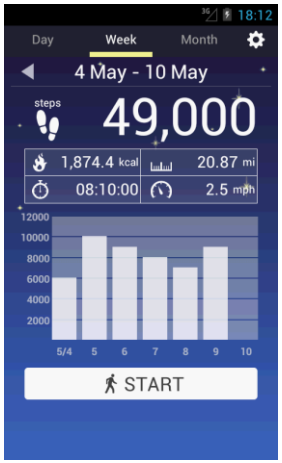
Sedentary Lifestyle

- Sedentary lifestyle
  - increases risk of diabetes, heart disease, dying earlier, etc
  - Kills more than smoking!!
- Categorization of sedentary lifestyle based on step count by paper:
  - "Catrine Tudor-Locke, Cora L. Craig, John P. Thyfault, and John C. Spence, A step-defined sedentary lifestyle index: < 5000 steps/day", Appl. Physiol. Nutr. Metab. 38: 100–114 (2013)



Step Count Mania

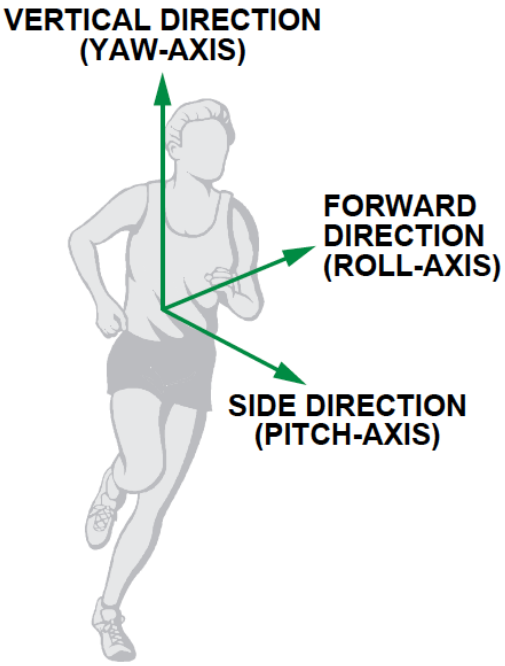
- Everyone is crazy about step count these days
- Pedometer apps, pedometers, fitness trackers, etc
- Tracking makes user aware of activity levels, motivates them to exercise more

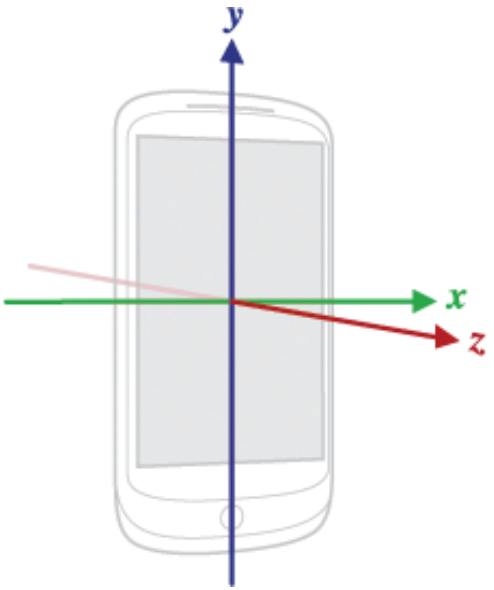


How does a Pedometer Detect/Count Steps

Ref: Deepak Ganesan, Ch 2 Designing a Pedometer and Calorie Counter

- As example of processing Accelerometer data
- Walking or running results in motion along the 3 body axes (forward, vertical, side)
- Smartphone has similar axes
  - Alignment depends on phone orientation

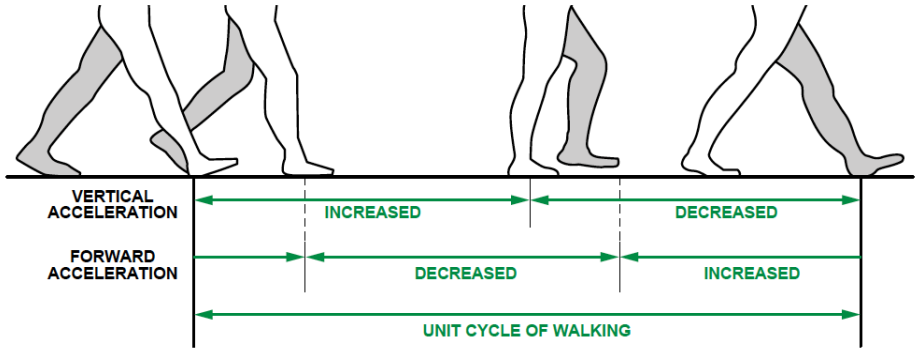


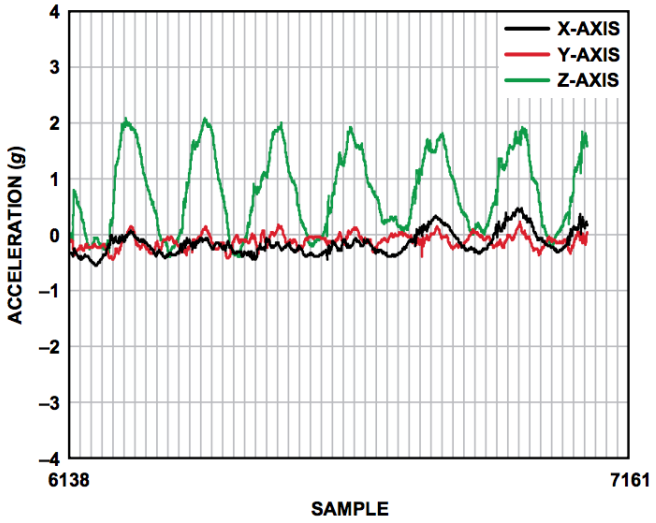


The Nature of Walking

Ref: Deepak Ganesan, Ch 2 Designing a Pedometer and Calorie Counter

- Vertical and forward acceleration increases/decreases during different phases of walking
- Walking causes a large periodic spike in one of the accelerometer axes
- Which axes (x, y or z) and magnitude depends on phone orientation

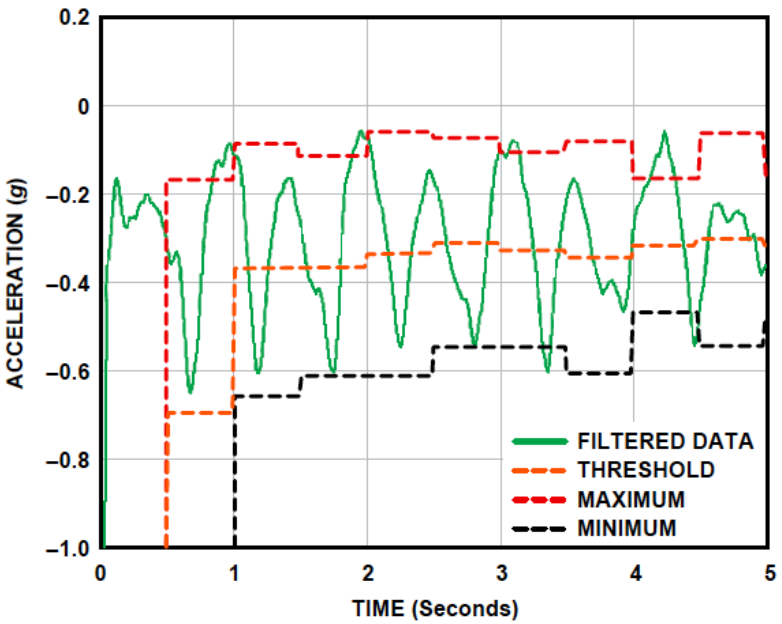




Step Detection Algorithm

Ref: Deepak Ganesan, Ch 2 Designing a Pedometer and Calorie Counter

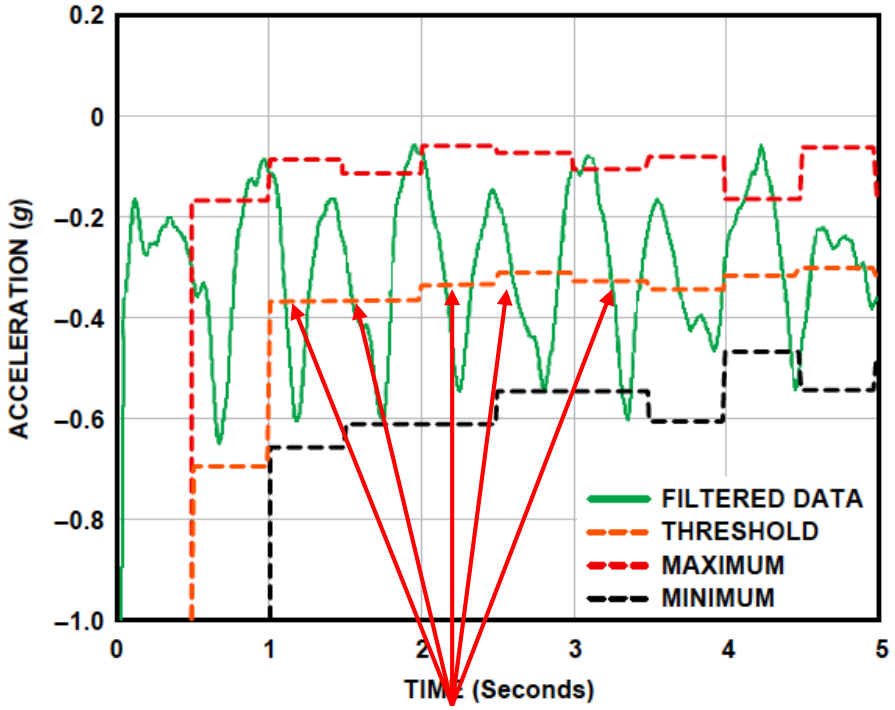
- **Step 1: smoothing**
  - Signal looks choppy
  - Smooth by replacing each sample with average of current, prior and next sample (Window of 3)
- **Step 2: Dynamic Threshold Detection**
  - Focus on accelerometer axis with largest peak
  - Would like a threshold such that each crossing is a step
  - But cannot assume fixed threshold (magnitude depends on phone orientation)
  - Track min, max values observed every 50 samples
  - Compute ***dynamic threshold: (Max + Min)/2***



Step Detection Algorithm

Ref: Deepak Ganesan, Ch 2 Designing a Pedometer and Calorie Counter

- A step is
  - indicated by crossings of dynamic threshold
  - Defined as negative slope (sample_new < sample_old) when smoothed waveform crosses dynamic threshold

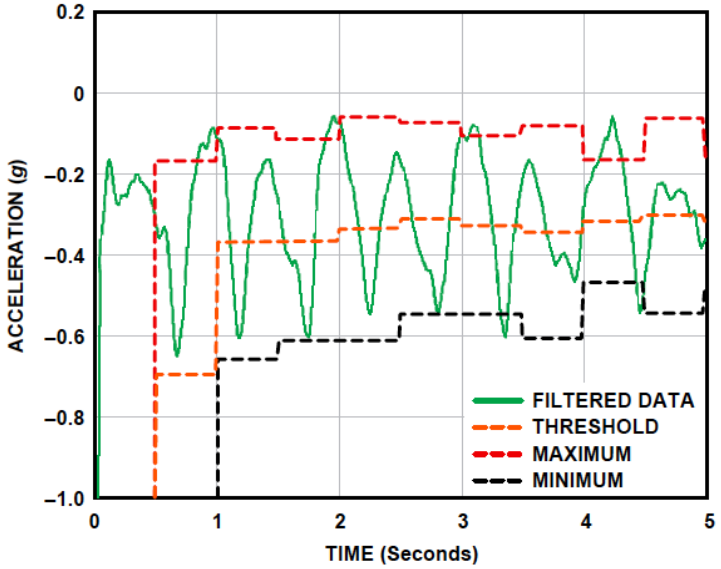


Steps

Step Detection Algorithms

Ref: Deepak Ganesan, Ch 2 Designing a Pedometer and Calorie Counter

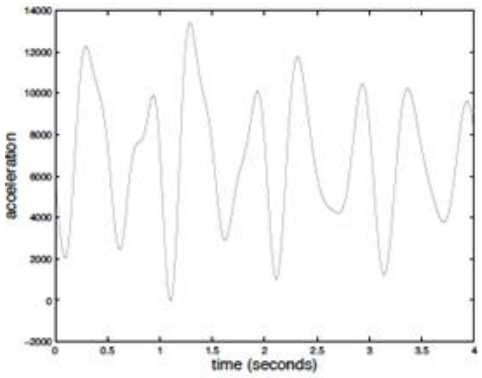
- **Problem:** vibrations (e.g. mowing lawn, plane taking off) could be counted as a step
- **Optimization:** Fix by exploiting periodicity of walking/running
- Assume people can:
  - Run 5 steps per second => 0.2 seconds per step
  - Walk 1 step every 2 seconds => 2 seconds per step
  - So, can eliminate "negative crossings" that occur outside period [0.2 – 2 seconds]



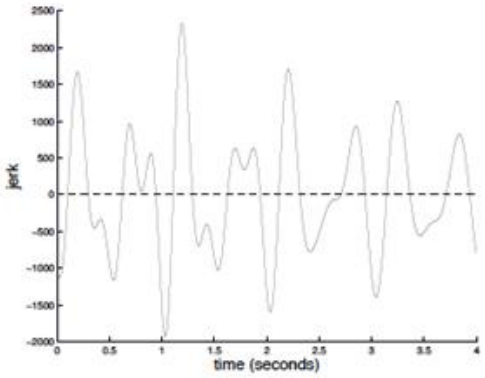
Step Detection Algorithms

Ref: Deepak Ganesan, Ch 2 Designing a Pedometer and Calorie Counter

- Previous step detection algorithm is simple.
- More sophisticated algorithms exist
- Smoothing: Time domain filtering
  - Exponential smoothing: Weight more recent samples higher
  - Median filtering + Exponential smoothing
- Frequency domain processing:
  - Fourier transform, operations in frequency domain
  - Keep frequencies of typical walking, and remove rest
  - Typical walking pace: 2-3Hz (remove freq > 5Hz)



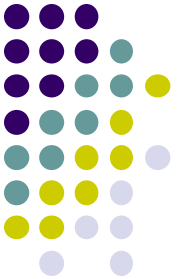
(c) Output of the low-pass filter.



(d) Derivative of the low-pass filter.

Counting Calories

Ref: Deepak Ganesan, Ch 2 Designing a Pedometer and Calorie Counter



- First, calculate distance covered based on number of steps taken

Distance = number of steps × distance per step (1)

- Distance per step (stride) depends on user's height (taller people, longer strides)
- Number of steps taken per 2 seconds gives estimate of person's stride length

Steps per 2 s	Stride (m/s)
0~2	Height/5
2~3	Height/4
3~4	Height/3
4~5	Height/2
5~6	Height/1.2
6~8	Height
>=8	1.2 × Height

Counting Calories

Ref: Deepak Ganesan, Ch 2 Designing a Pedometer and Calorie Counter

- To estimate speed, remember that speed = distance/time. Thus,

Speed = steps per 2 s × stride/2 s (2)

- Many factors affect calorie expenditure. E.g
  - Body weight, workout intensity, fitness level, etc
- Rough relationship given in table

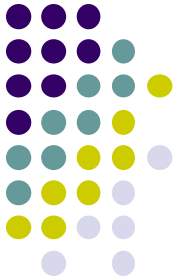
Running Speed (km/h)	Calories Expended (C/kg/h)
8	10
12	15
16	20
20	25

- Expressed as an equation

Calories (C/kg/h) = 1.25 × running speed (km/h) (3)

- Converting from speed in km/h to m/s

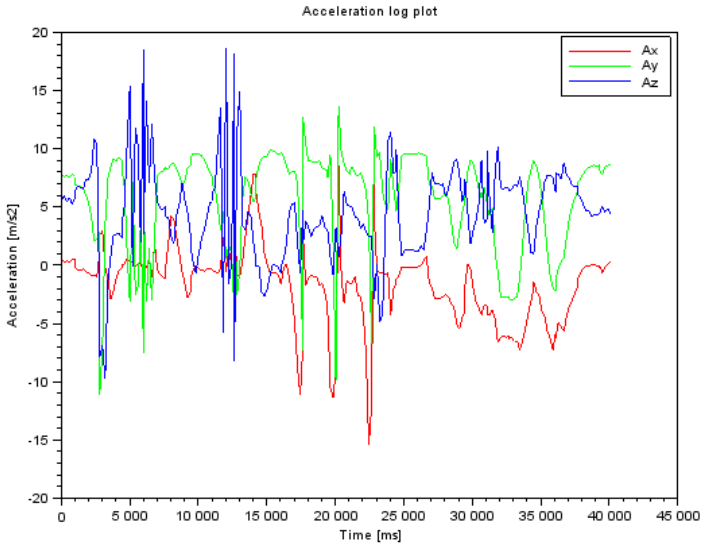
Calories (C/kg/h) = 1.25 × speed (m/s) × 3600/1000 = 4.5 × speed (m/s) (4)



Introduction to Activity Recognition

Activity Recognition

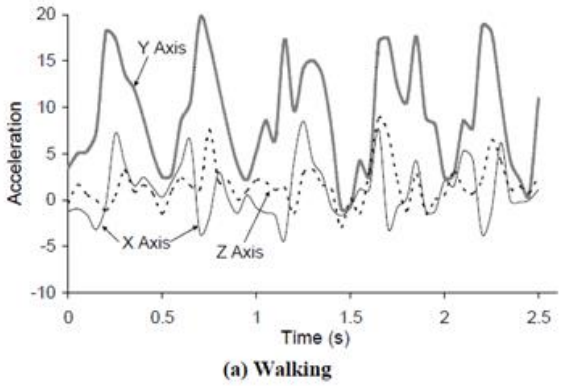
- **Goal:** Want our app to detect what activity the user is doing?
- **Classification task:** which of these 6 activities is user doing?
  - Walking,
  - Jogging,
  - Ascending stairs,
  - Descending stairs,
  - Sitting,
  - Standing



- Typically, use machine learning classifers to classify user's accelerometer signals

Activity Recognition Overview

Gather Accelerometer data



(a) Walking

Machine Learning Classifier

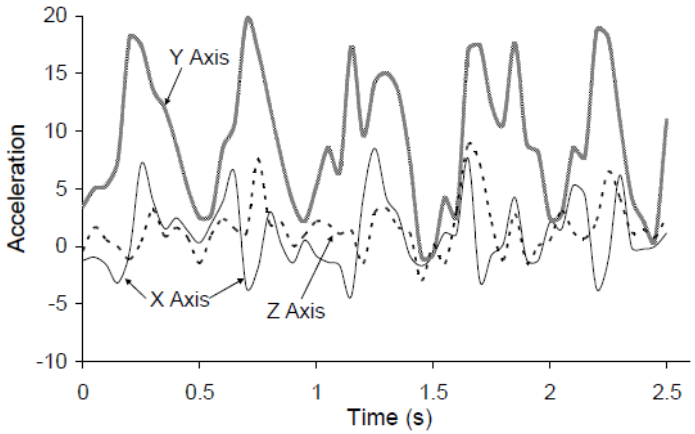
Classify Accelerometer data

Walking

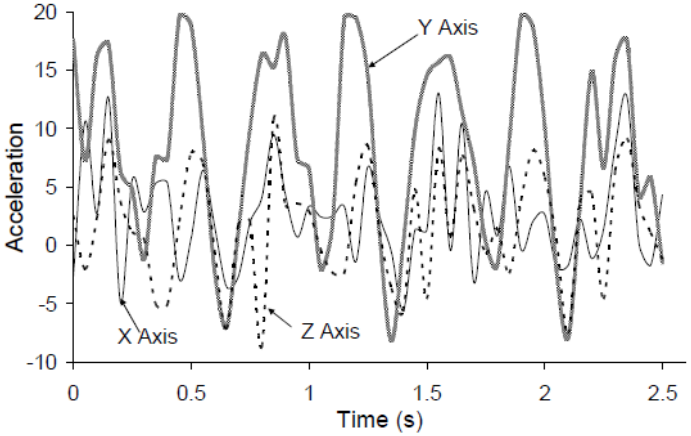
Running

Climbing Stairs

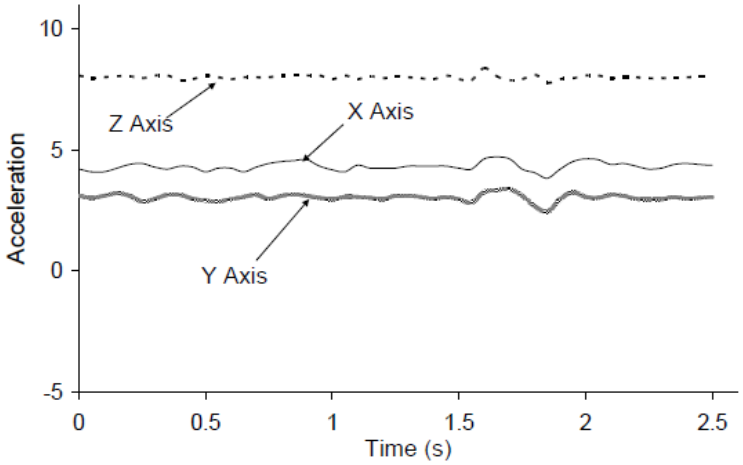
Example Accelerometer Data for Activities



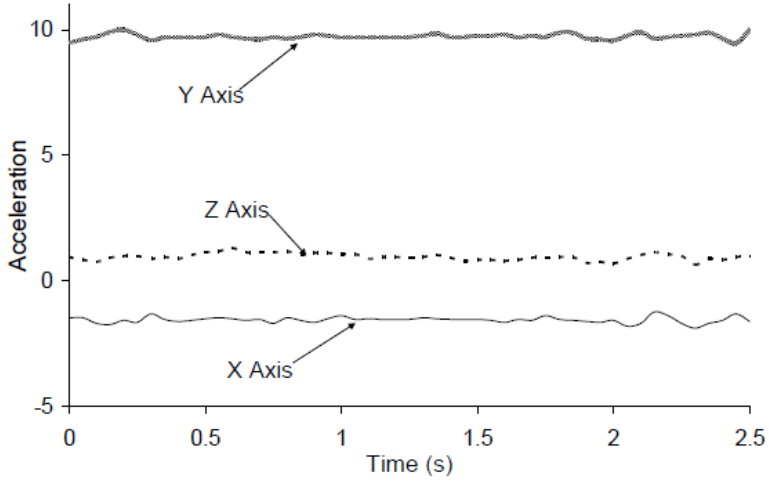
(a) Walking



(b) Jogging

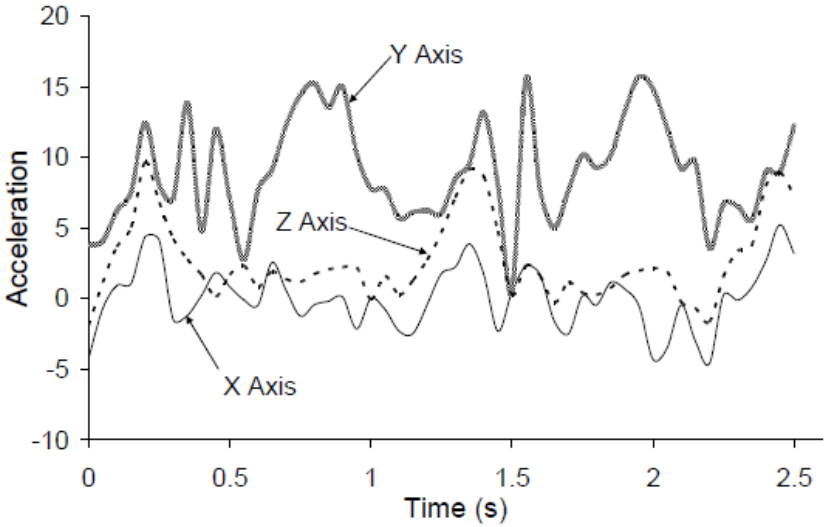


(e) Sitting

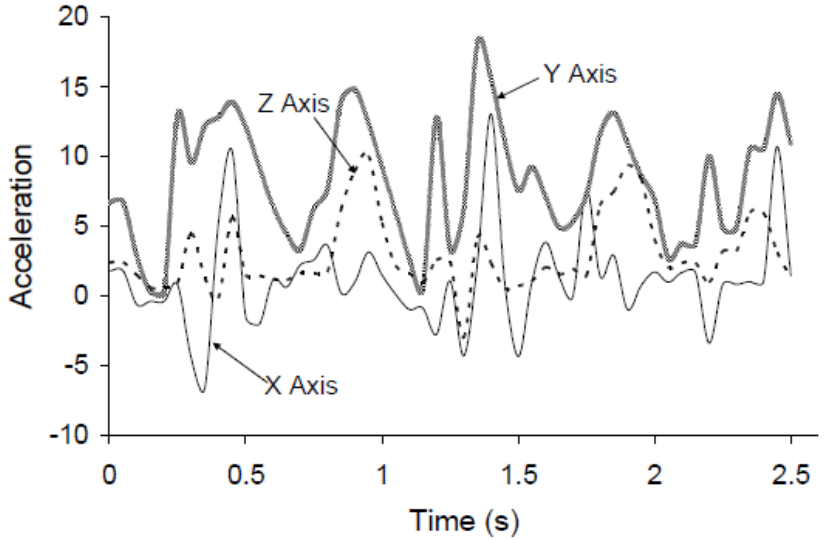


(f) Standing

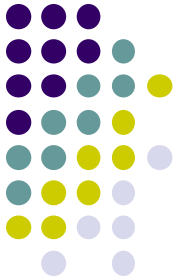
Example Accelerometer Data for Activities



(c) Ascending Stairs



(d) Descending Stairs



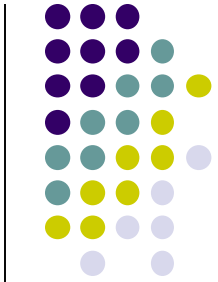
Activity Recognition Using Google API

Activity Recognition

- Activity Recognition? Detect what user is doing?
  - Part of user's context
- Examples: sitting, running, driving, walking
- Why? App can adapt it's behavior based on user behavior
- **E.g.** If user is driving, don't send notifications

https://www.youtube.com/watch?v=S8sugXgUVEI

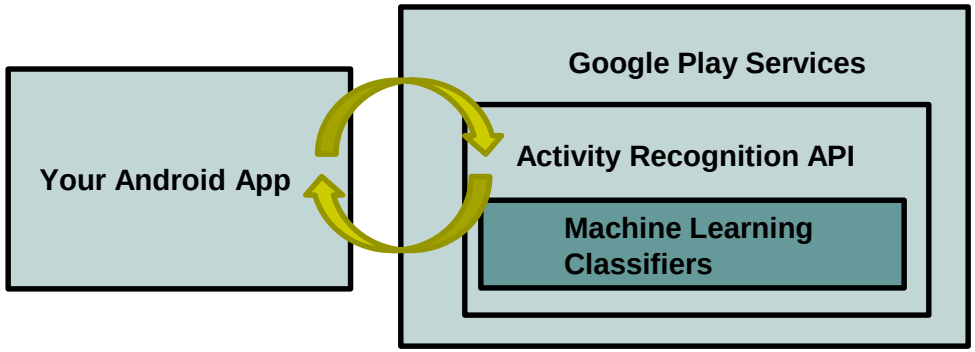
Google Activity Recognition API



- API to detect smartphone user's current activity
- Programmable, can be used by your Android app
- Currently detects 6 states:
  - In vehicle
  - On Bicycle
  - On Foot
  - Still
  - Tilting
  - Unknown

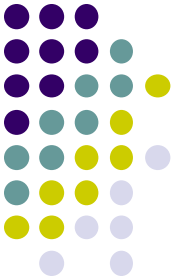
Google Activity Recognition API

- Deployed as part of Google Play Services



Activity Recognition Using AR API

Ref: How to Recognize User Activity with Activity Recognition by Paul Trebilcox-Ruiz on Tutsplus.com tutorials



- Example code for this tutorial on gitHub:

  https://github.com/tutsplus/Android-ActivityRecognition

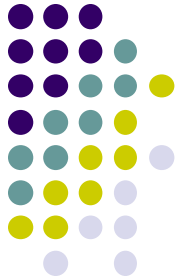
- Google Activity Recognition can:
  - Recognize user's current activity (Running, walking, in a vehicle or still)

- Project Setup:
  - Create Android Studio project with blank Activity (minimum SDK 14)
  - In **build.gradle** file, define latest Google Play services (was 8.4 last year, now 11.5.9) as dependency

```
compile 'com.google.android.gms:play-services:8.4.0'
```

Activity Recognition Using AR API

Ref: How to Recognize User Activity with Activity Recognition by Paul Trebilcox-Ruiz on Tutsplus.com tutorials



- Create new class **ActivityRecognizedService** which extends **IntentService**
- **IntentService:** type of service, asynchronously handles work off main thread as Intent requests.
- Throughout user's day, **Activity Recognition API** sends user's activity to this IntentService in the background
- Need to program this Intent to handle incoming user activity

```
01  public class ActivityRecognizedService extends IntentService {
02
03      public ActivityRecognizedService() {
04          super("ActivityRecognizedService");
05      }
06
07      public ActivityRecognizedService(String name) {
08          super(name);
09      }
10
11      @Override
12      protected void onHandleIntent(Intent intent) {
13      }
14  }
```

Called by Android OS to deliver User's activity

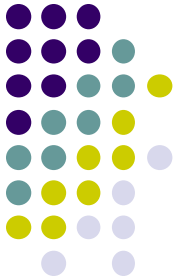
Activity Recognition Using AR API

Ref: How to Recognize User Activity with Activity Recognition by Paul Trebilcox-Ruiz on Tutsplus.com tutorials

- Modify **AndroidManifest.xml** to
  - Declare **ActivityRecognizedService**
  - Add com.google.android.gms.permission.ACTIVITY_RECOGNITION permission

```
01<?xml version="1.0" encoding="utf-8"?>
02<manifest xmlns:android="http://schemas.android.com/apk/res/android"
03    package="com.tutsplus.activityrecognition">
04
05    <uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />
06
07    <application
08        android:icon="@mipmap/ic_launcher"
09        android:label="@string/app_name"
10        android:theme="@style/AppTheme">
11        <activity android:name=".MainActivity">
12            <intent-filter>
13                <action android:name="android.intent.action.MAIN" />
14
15                <category android:name="android.intent.category.LAUNCHER" />
16            </intent-filter>
17        </activity>
18
19        <service android:name=".ActivityRecognizedService" />
20    </application>
21
22</manifest>
```

Requesting Activity Recognition



- In **MainActivity.java**, To connect to Google Play Services:
  - Provide **GoogleApiClient** variable type + implement callbacks

```
01 public class MainActivity extends AppCompatActivity implements GoogleApiClient.ConnectionCallbacks,
02 GoogleApiClient.OnConnectionFailedListener {
03
04     public GoogleApiClient mApiClient;
05
06     @Override
07     protected void onCreate(Bundle savedInstanceState) {
08         super.onCreate(savedInstanceState);
09         setContentView(R.layout.activity_main);
10     }
11
12     @Override
13     public void onConnected(@Nullable Bundle bundle) {
14
15     }
16
17     @Override
18     public void onConnectionSuspended(int i) {
19
20     }
21
22     @Override
23     public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
24
25     }
}
```

Handle to Google Activity Recognition client

Normal AR call if everything working well

Called if sensor (accelerometer) connection fails

Called if Google Play connection fails

Requesting Activity Recognition

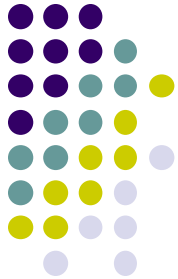
- In onCreate, initialize client and connect to Google Play Services

```
01  @Override
02  protected void onCreate(Bundle savedInstanceState) {
03      super.onCreate(savedInstanceState);
04      setContentView(R.layout.activity_main);
05
06      mApiClient = new GoogleApiClient.Builder(this)
07              .addApi(ActivityRecognition.API)
08              .addConnectionCallbacks(this)
09              .addOnConnectionFailedListener(this)
10              .build();
11
12      mApiClient.connect();
13  }
```

Request ActivityRecognition.API

Associate listeners with our instance of GoogleApiClient

Requesting Activity Recognition



- Once **GoogleApiClient** has connected, **onConnected( )** is called
- Need to create a **PendingIntent** that goes to our **IntentService**
- Also set how often API should check user's activity in milliseconds

```
1@Override
2public void onConnected(@Nullable Bundle bundle) {
3    Intent intent = new Intent( this, ActivityRecognizedService.class );
4    PendingIntent pendingIntent = PendingIntent.getService( this, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT );
5    ActivityRecognition.ActivityRecognitionApi.requestActivityUpdates( mApiClient, 3000, pendingIntent );
6}
```

Build intent to send to IntentService

How often to check user's activity (in milliseconds)

Handling Activity Recognition

- Our app tries to recognize the user's activity every 3 seconds
- **onHandleIntent** called every 3 seconds, Intent delivered
- In **onHandleIntent( )** method of **ActivityRecognizedService**
  - Extract **ActivityRecognitionResult** from the Intent
  - Retrieve list of possible activities by calling **getProbableActivities( )** on **ActivityRecognitionResult** object

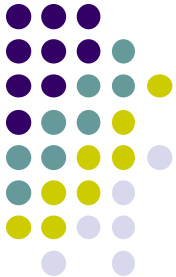
```
1@Override
2protected void onHandleIntent(Intent intent) {
3   if(ActivityRecognitionResult.hasResult(intent)) {
4      ActivityRecognitionResult result = ActivityRecognitionResult.extractResult(intent);
5      handleDetectedActivities( result.getProbableActivities() );
6   }
7}
```

Called to deliver user's activity as an Intent

Extract Activity Recognition object from Intent

Get list of probable activities

Handling Activity Recognition



- Simply log each detected activity and display how confident Google Play services is that user is performing this activity

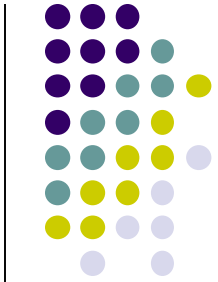
```
private void handleDetectedActivities(List<DetectedActivity> probableActivities) {
    for( DetectedActivity activity : probableActivities ) {
        switch( activity.getType() ) {
            case DetectedActivity.IN_VEHICLE: {
                Log.e( "ActivityRecogition", "In Vehicle: " + activity.getConfidence() );
                break;
            }
            case DetectedActivity.ON_BICYCLE: {
                Log.e( "ActivityRecogition", "On Bicycle: " + activity.getConfidence() );
                break;
            }
            case DetectedActivity.ON_FOOT: {
                Log.e( "ActivityRecogition", "On Foot: " + activity.getConfidence() );
                break;
            }
            case DetectedActivity.RUNNING: {
                Log.e( "ActivityRecogition", "Running: " + activity.getConfidence() );
                break;
            }
            case DetectedActivity.STILL: {
                Log.e( "ActivityRecogition", "Still: " + activity.getConfidence() );
                break;
            }
            case DetectedActivity.TILTING: {
                Log.e( "ActivityRecogition", "Tilting: " + activity.getConfidence() );
                break;
            }
```

Switch statement on activity type

Sample output

```
1  E/ActivityRecogition: On Foot: 92
2  E/ActivityRecogition: Running: 87
3  E/ActivityRecogition: On Bicycle: 8
4  E/ActivityRecogition: Walking: 5
```

Handling Activity Recognition



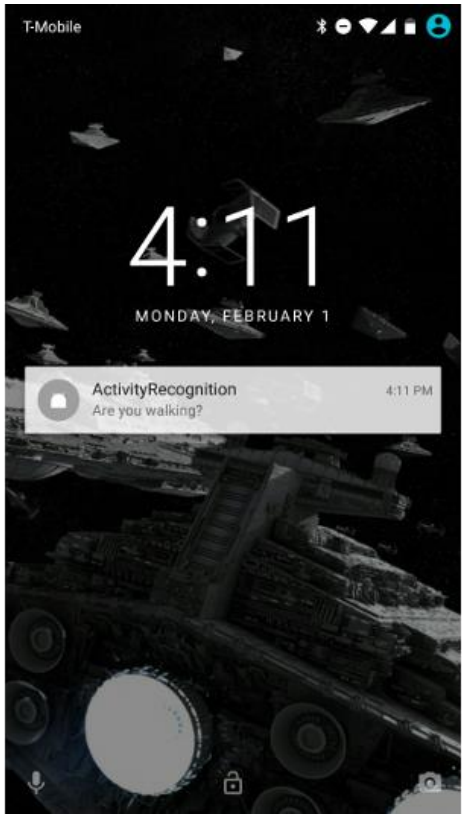
- If confidence is > 75, activity detection is probably accurate
- If user is walking, ask “Are you walking?”

```
            case DetectedActivity.WALKING: {
                Log.e( "ActivityRecogition", "Walking: " + activity.getConfidence() );
                if( activity.getConfidence() >= 75 ) {
                    NotificationCompat.Builder builder = new NotificationCompat.Builder(this);
                    builder.setContentText( "Are you walking?" );
                    builder.setSmallIcon( R.mipmap.ic_launcher );
                    builder.setContentTitle( getString( R.string.app_name ) );
                    NotificationManagerCompat.from(this).notify(0, builder.build());
                }
                break;
            }
            case DetectedActivity.UNKNOWN: {
                Log.e( "ActivityRecogition", "Unknown: " + activity.getConfidence() );
                break;
            }
        }
    }
}
```

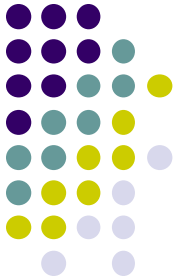
Sample Output of Program

- Sample displayed on development console

```
1  E/ActivityRecogition: On Foot: 92
2  E/ActivityRecogition: Running: 87
3  E/ActivityRecogition: On Bicycle: 8
4  E/ActivityRecogition: Walking: 5
```



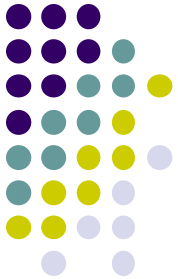
- Full code at: https://github.com/tutsplus/Android-ActivityRecognition



Android Awareness API

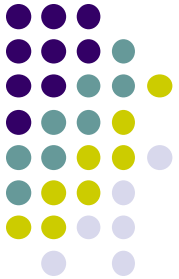
Awareness API

https://developers.google.com/awareness/overview



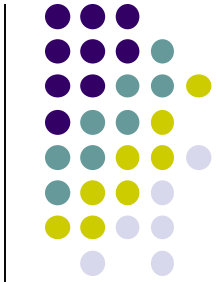
- Single Android API for context awareness released in 2016
- Combines some APIs already covered (Place, Activity, Location)

Context type	Example
Time	Current local time
Location	Latitude and longitude
Place	Place, including place type
Activity	Detected user activity (walking, running, biking)
Beacons	Nearby beacons matching the specified namespace
Headphones	Are headphones plugged in?
Weather	Current weather conditions



Quiz 3

Quiz 3



- Quiz in class next Thursday (before class Oct 12)
- Short answer questions
- Try to focus on understanding, not memorization
- Covers:
  - Lecture slides for lectures 5a,5b,6a,6b
  - 1 code example from book
    - **HFAD examples:** Odometer (Distance Travelled), Ch 13. pg 541

References

- Head First Android
- Android Nerd Ranch, 2nd edition
- Busy Coder's guide to Android version 6.3
- CS 65/165 slides, Dartmouth College, Spring 2014
- CS 371M slides, U of Texas Austin, Spring 2014