CS 528 Mobile and Ubiquitous Computing

Lecture 5b: Step Counting & Activity Recognition

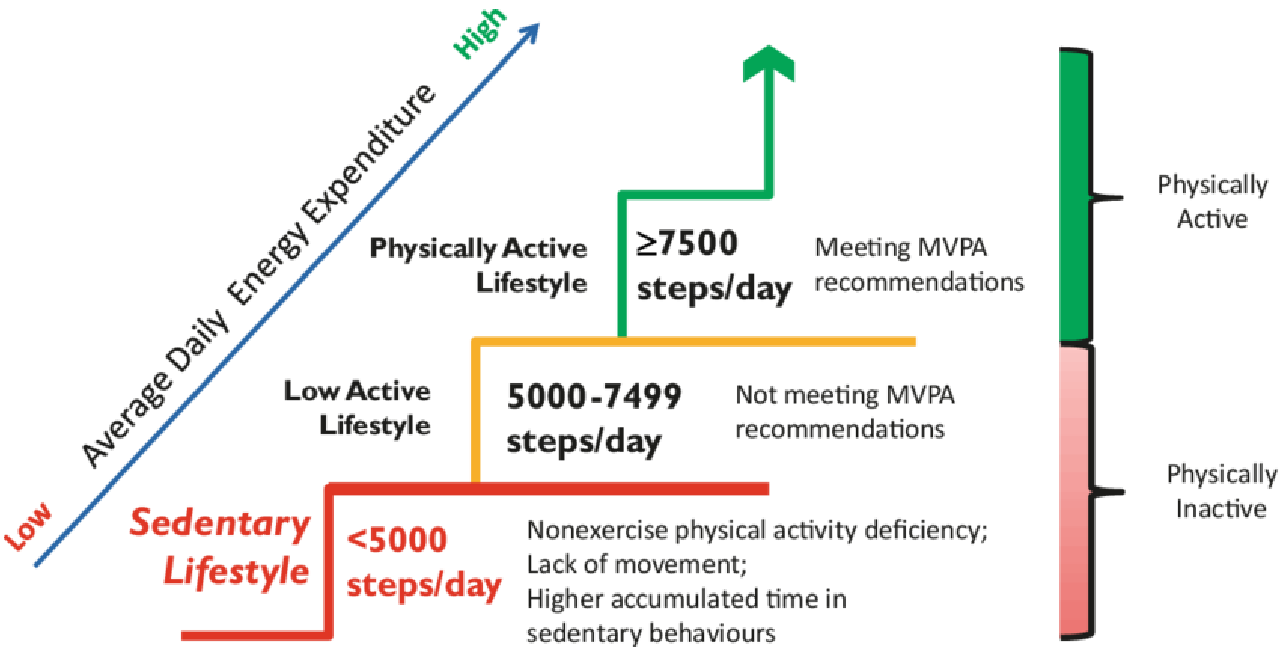
Emmanuel Agu

Step Counting

(How Step Counting Works)

Sedentary Lifestyle

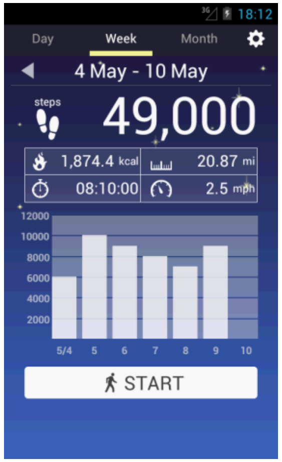
- Sedentary lifestyle
  - increases risk of diabetes, heart disease, dying earlier, etc
  - Kills more than smoking!!
- Categorization of sedentary lifestyle based on step count by paper:
  - "Catrine Tudor-Locke, Cora L. Craig, John P. Thyfault, and John C. Spence, A step-defined sedentary lifestyle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



High
Average Daily Energy Expenditure
Low
Physically Active Lifestyle
≥7500 steps/day
Meeting MVPA recommendations
Low Active Lifestyle
5000-7499 steps/day
Not meeting MVPA recommendations
Sedentary Lifestyle
<5000 steps/day
Nonexercise physical activity deficiency;
Lack of movement;
Higher accumulated time in sedentary behaviours
Physically Active
Physically Inactive

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


18:12

Day Week Month

4 May - 10 May

steps 49,000

1,874.4 kcal 20.87 mi

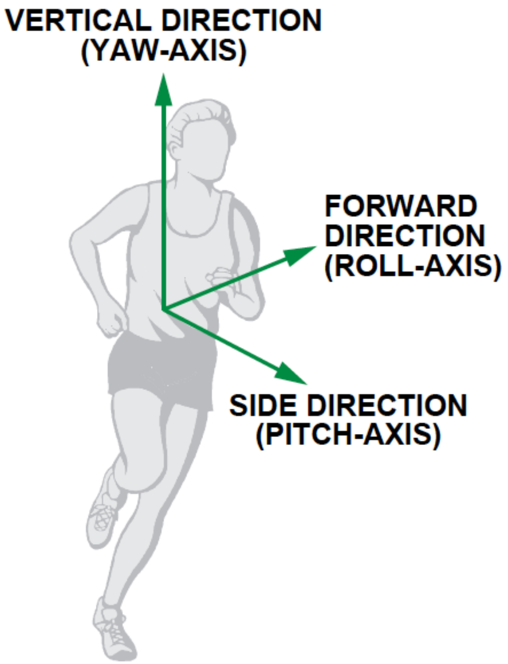
08:10:00 2.5 mph

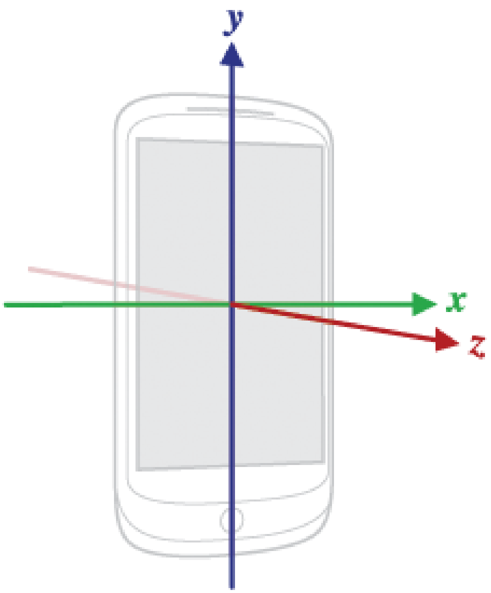
START

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

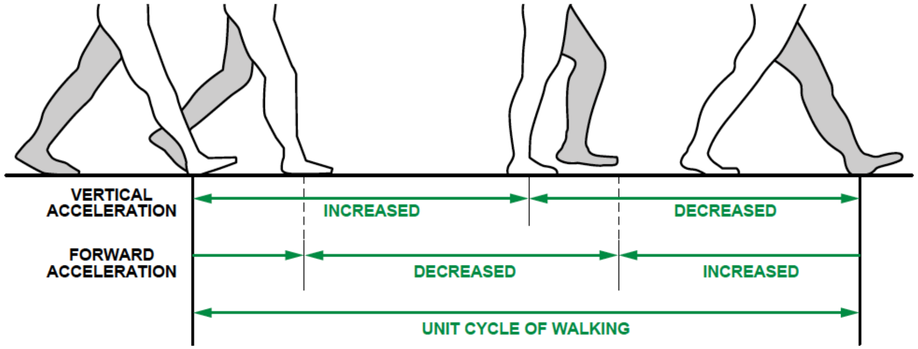


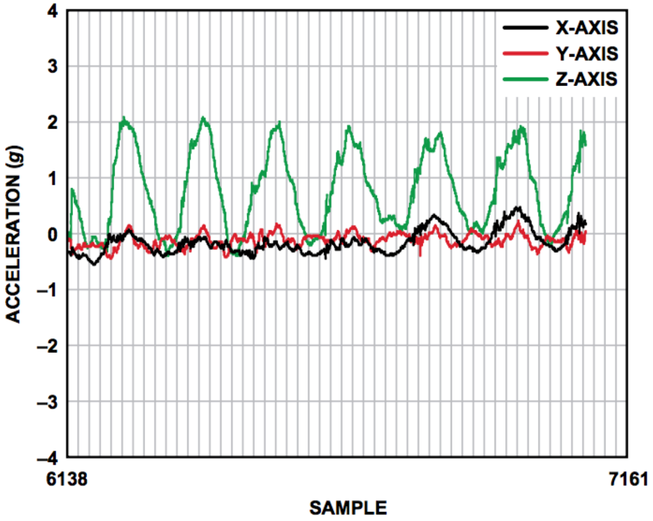


The Nature of Walking

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

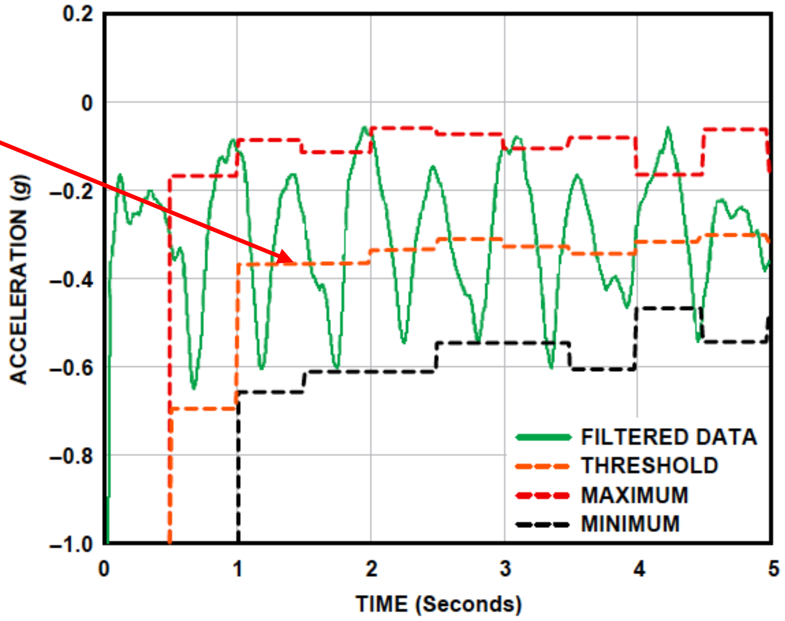




Step Detection Algorithm

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

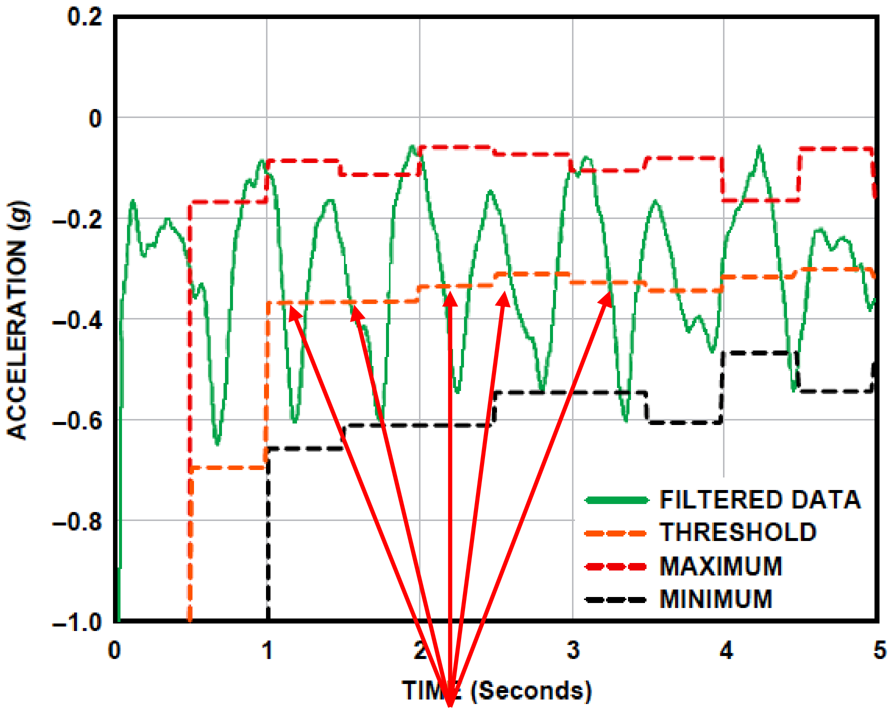
- **Step 1: smoothing**
  - Signal looks choppy
  - Smooth by replacing each sample with average of current, prior and next sample (Window of 3)
- **Step 2: Dynamic Threshold Detection**
  - Focus on accelerometer axis with largest peak
  - Would like a threshold such that each crossing is a step
  - 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

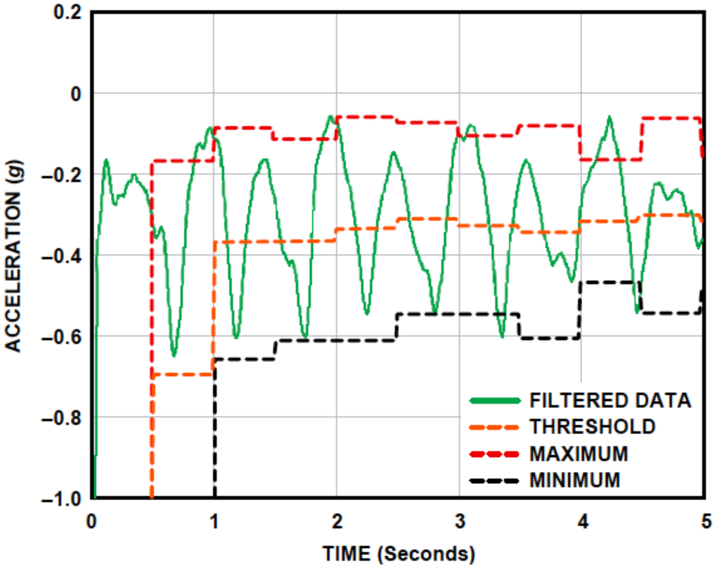


Steps

Step Detection Algorithms

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

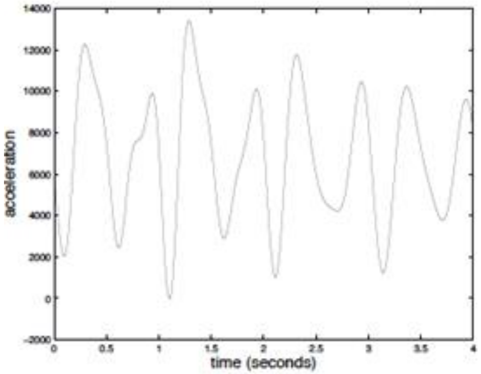
- **Problem:** vibrations (e.g. mowing lawn, plane taking off) could be counted as a step
- **Optimization:** Fix by exploiting periodicity of walking/running
- Assume people can:
  - **Run:** 5 steps per second => 0.2 seconds per step
  - **Walk:** 1 step every 2 seconds => 2 seconds per step
  - So,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



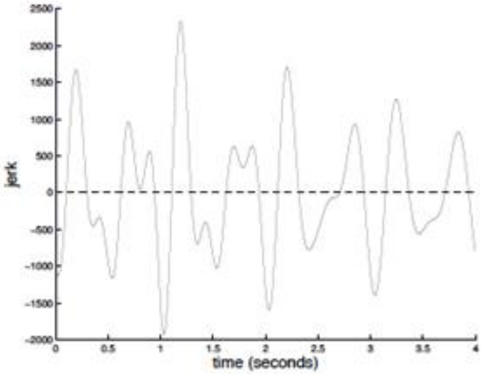
Step Detection Algorithms

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

- Previous step detection algorithm is simple.
- Can use more sophisticated signal processing algorithms for smoothing
- Frequency domain processing (E.g. Fourier transform + low-pass filter)



(c) Output of the low-pass filter.



(d) Derivative of the low-pass filter.

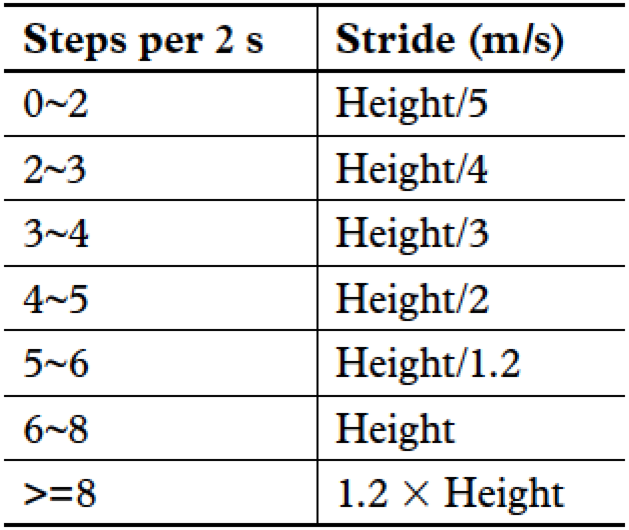
Estimate Distance Traveled

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

- 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)
- Using person's height, can estimate their stride, then number of steps taken per 2 seconds



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

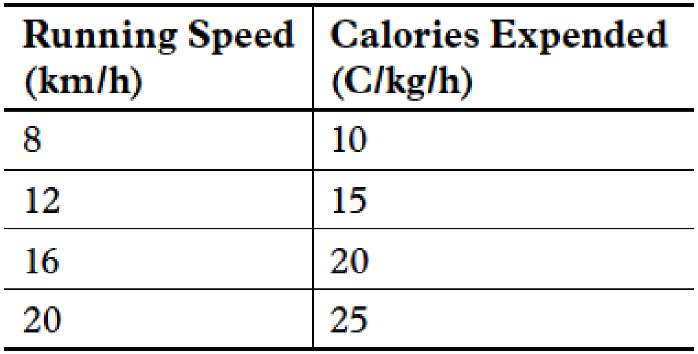
Estimating Calories Burned

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

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

Speed (in m/s) = (no. steps per 2 s × stride (in meters))/2s (2)

- Can also convert to calorie expenditure, which depends on many factors 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

x / y = 1.25

- Expressed as an equation

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

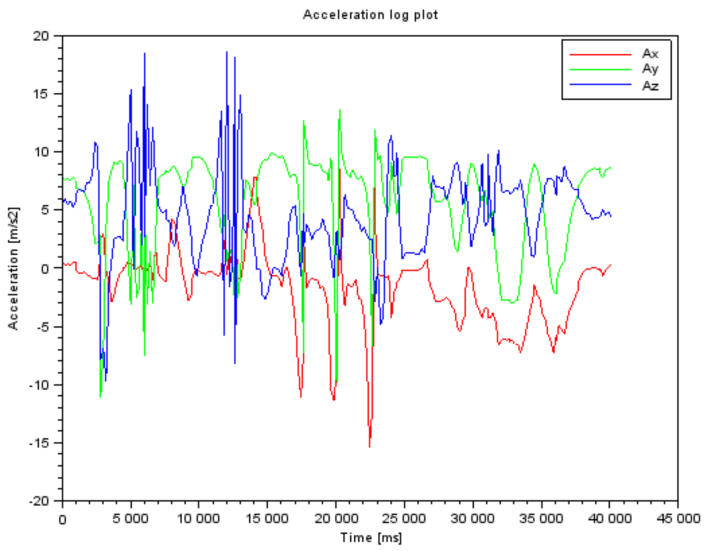
- First convert from speed in km/h to m/s

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

Introduction to Activity Recognition

Activity Recognition

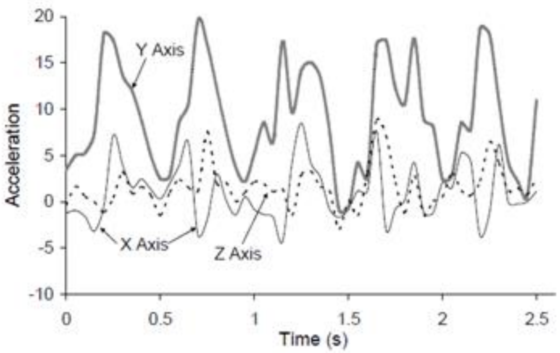
- **Goal:** Want our app to detect what activity the user is doing?
- **Classification task:** which of these 6 activities is user doing?
  - Walking,
  - Jogging,
  - Ascending stairs,
  - Descending stairs,
  - Sitting,
  - Standing



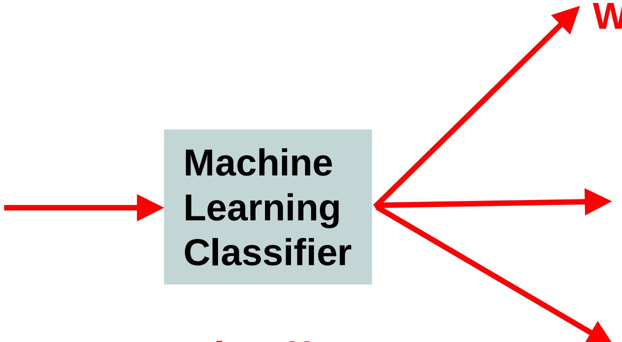
- Typically, use machine learning classifers to classify user’s accelerometer signals

Activity Recognition Overview

Gather Accelerometer data



(a) Walking



Machine Learning Classifier

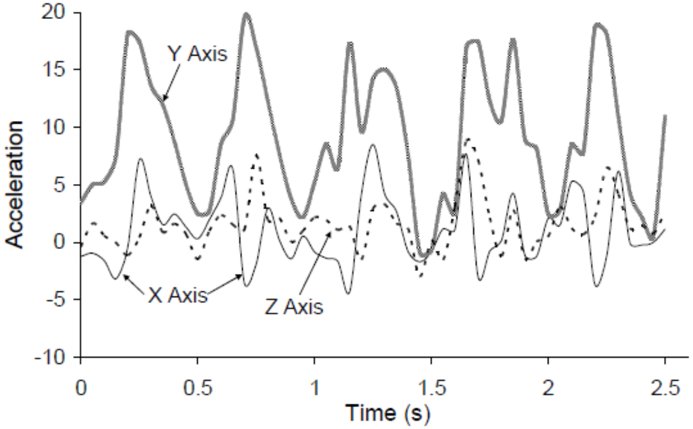
Walking

Running

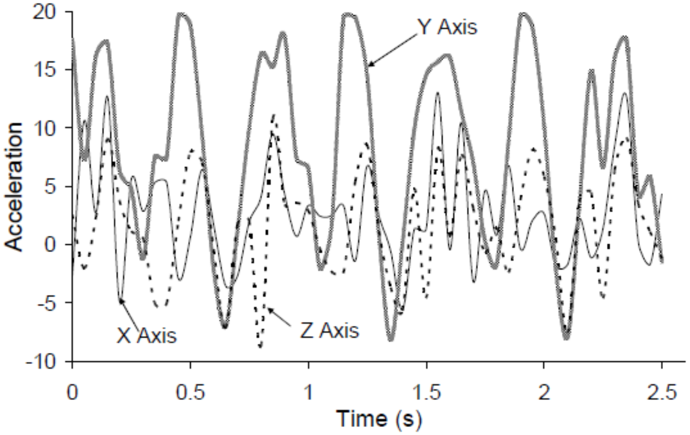
Climbing Stairs

Classify Accelerometer data

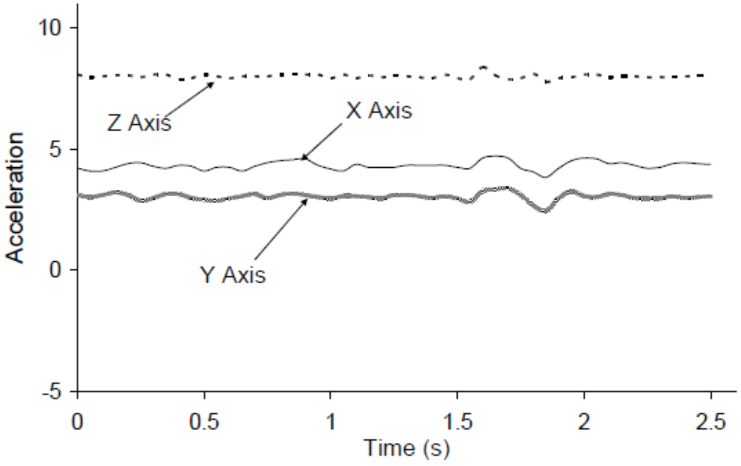
Example Accelerometer Data for Activities



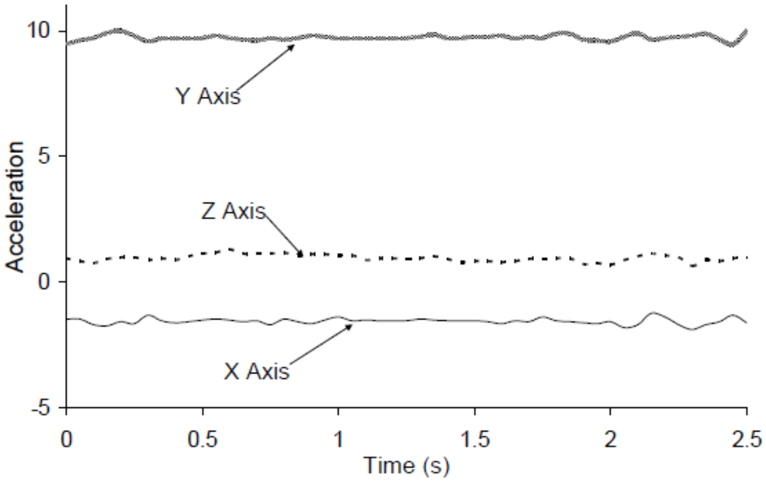
(a) Walking



(b) Jogging

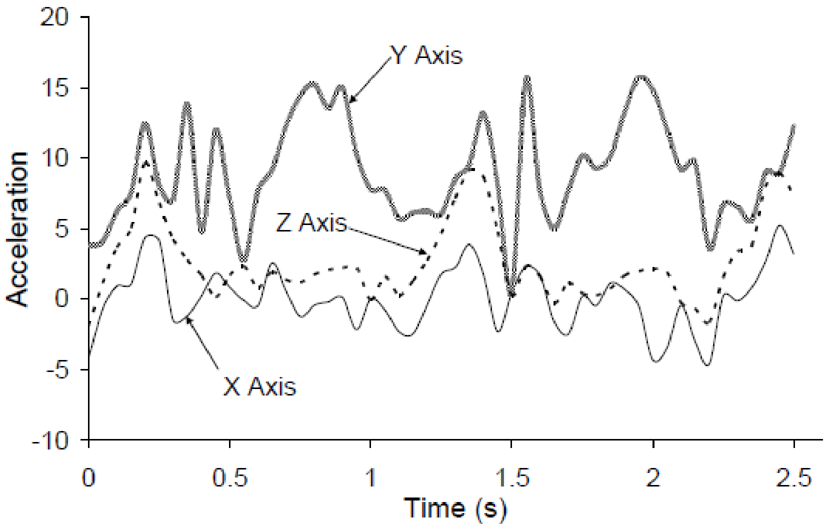


(e) Sitting

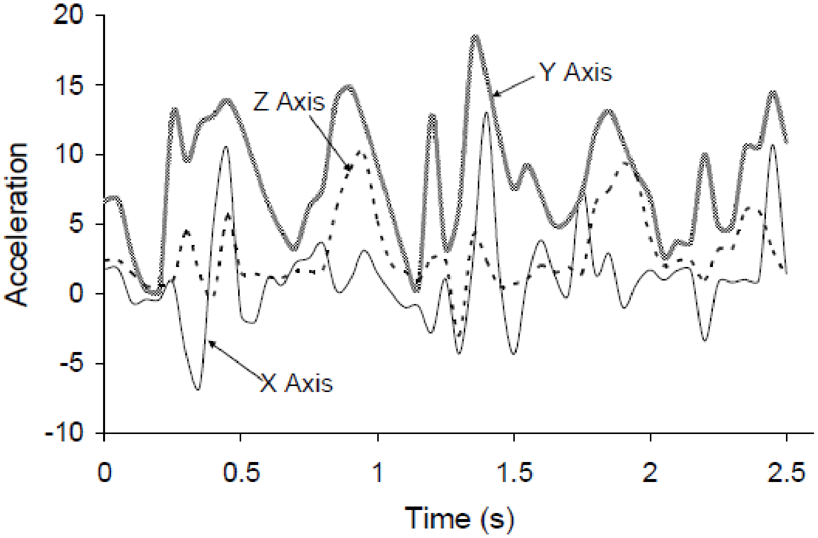


(f) Standing

Example Accelerometer Data for Activities



(c) Ascending Stairs

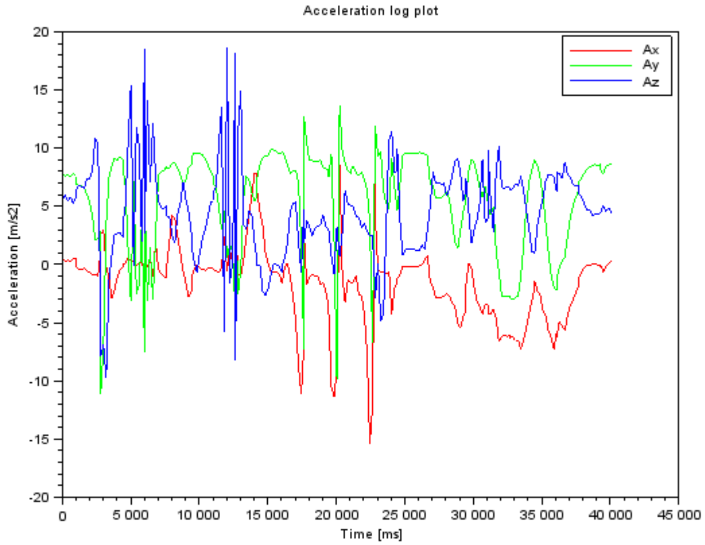


(d) Descending Stairs

Applications of Activity Recognition

Recall: 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

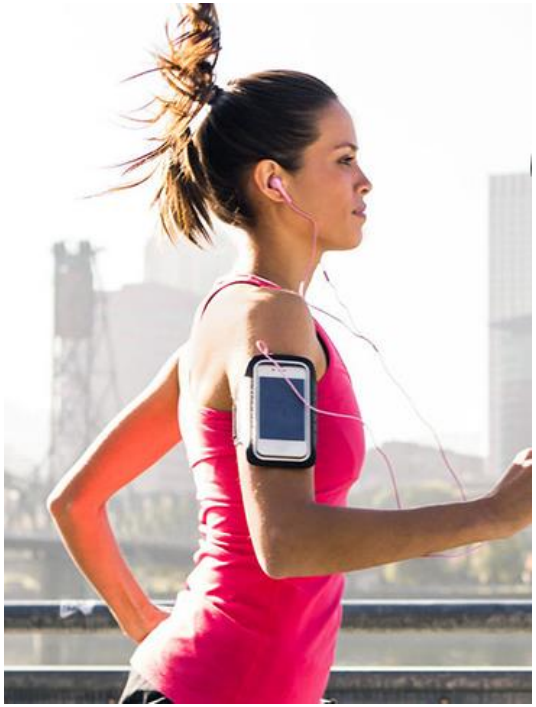


- Typically, use machine learning classifers to classify user’s accelerometer signals

Applications of Activity Recognition (AR)

Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Fitness Tracking:**
  - **Initially:**
    - Physical activity type,
    - Distance travelled,
    - Calories burned
  - **Newer features:**
    - Stairs climbed,
    - Physical activity (duration + intensity)
    - Activity type logging + context e.g. Ran 0.54 miles/hr faster during morning runs
    - Sleep tracking
    - Activity history



Note: AR refers to algorithm
But could run on a range of devices (smartphones, wearables, e.g. fitbit)

Applications of Activity Recognition (AR)

Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Health monitoring:** How **well** is patient performing activity?
- Make clinical monitoring pervasive, continuous, real world!!
  - Gather context information (e.g. what makes condition worse/better?)
  - E.g. timed up and go test
- Show patient contexts that worsen condition => Change behavior
  - E.g. walking in narror hallways worsens gait freeze

Parkinsons disease
Gait freezing

Question: What data would you need to build PD gait classifier? From what types of subjects?

COPD, Walk tests in the wild

Applications of Activity Recognition

Ref: Lockhart *et al*, Applications of Mobile Activity recognition

- **Fall:** Leading cause of death for seniors
- **Fall detection:** Smartphone/watch, wearable detects senior who has fallen, alert family
  - Text message, email, call relative

Fall detection + prediction

Applications of Activity Recognition (AR)

Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Context-Aware Behavior:**
  - In-meeting? => Phone switches to silent mode
  - Exercising? => Play song from playlist, use larger font sizes for text
  - Arrived at work? => download email
- Study found that messages delivered when transitioning between activities better received
- **Adaptive Systems to Improve User Experience:**
  - Walking, running, riding bike? => Turn off Bluetooth, WiFi (save power)
  - Can increase battery life up to 5x

Applications of AR

Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Smart home:**
  - Determine what activities people in the home are doing,
    - **Why?** infer illness, wellness, patterns, intrusion (security), etc
    - E.g. TV automatically turns on at about when you usually lie on the couch

Applications of AR: 3rd Party Apps

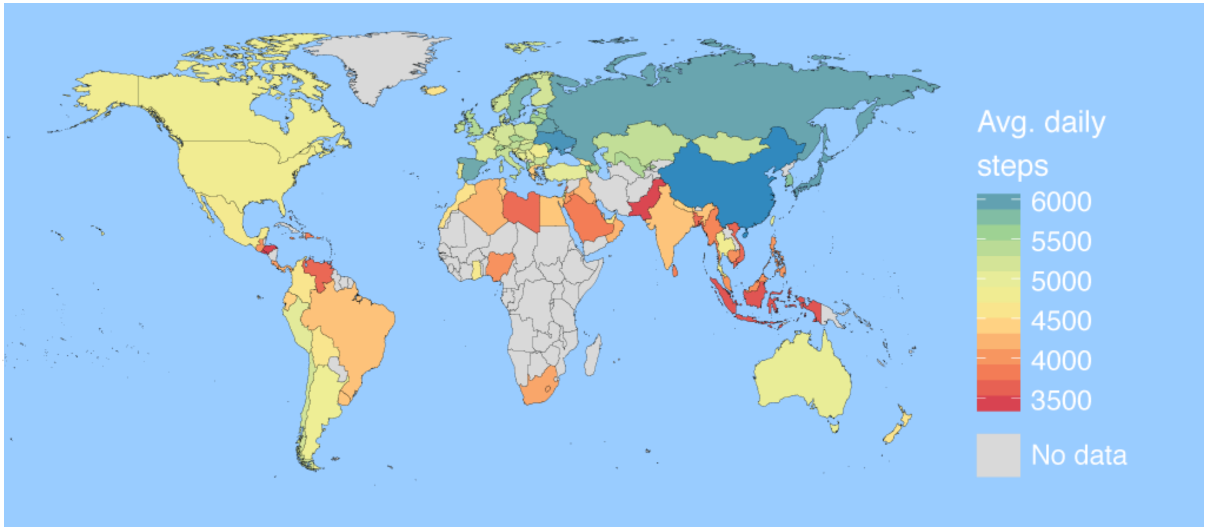
Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Targeted Advertising:**
  - AR helps deliver more relevant ads
  - E.g user runs a lot => Get exercise clothing ads
  - Goes to pizza places often + sits there => Get pizza ads

Applications of AR: 3rd Party Apps

Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Research Platforms for Data Collection:**
  - E.g. public health officials want to know how much time various people (e.g. students) spend sleeping, walking, exercising, etc
  - Mobile AR: inexpensive, automated data collection
  - E.g. Stanford Inequality project: Analyzed physical activity of 700k users in 111 countries using smartphone AR data
  - http://activityinequality.stanford.edu/



Applications of AR: 3rd Party Apps

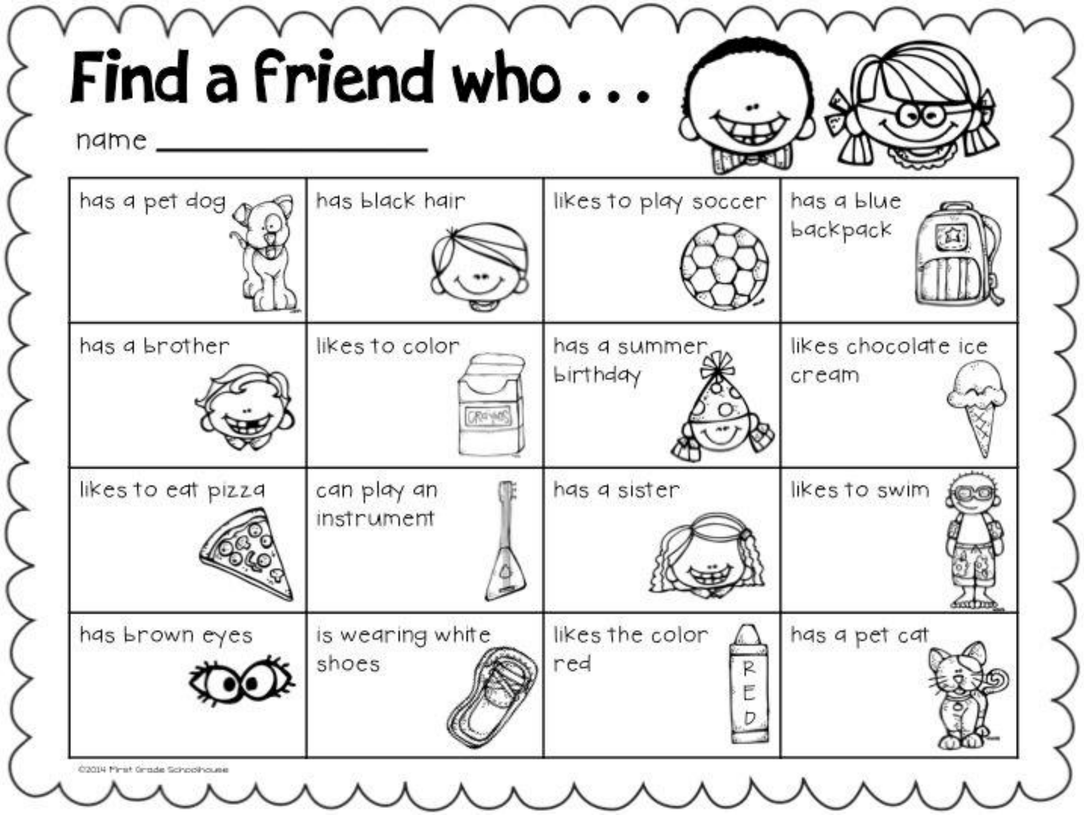
Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Track, manage staff on-demand:**
  - E.g. at hospital, determine “availability of nurses”, assign them to new jobs/patients/surgeries/cases

Applications of AR: Social Networking

Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Activity-Based Social Networking:**
  - Automatically connect users who do same activities + live close together



Find a friend who . . .

name ____________

has a pet dog	has black hair	likes to play soccer	has a blue backpack
has a brother	likes to color	has a summer birthday	likes chocolate ice cream
likes to eat pizza	can play an instrument	has a sister	likes to swim
has brown eyes	is wearing white shoes	likes the color red	has a pet cat

©2014 First Grade Schoolhouse

Applications of AR: Social Networking

Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Activity-Based Place Tagging:**
  - Automatically "popular" places where users perform same activity
  - E.g. Park street is popular for runners (activity-based maps)

- **Automatic Status updates:**
  - E.g. Bob is sleeping
  - Tracy is jogging along Broadway with track team
  - Privacy/security concerns => Different Levels of details for different friends

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

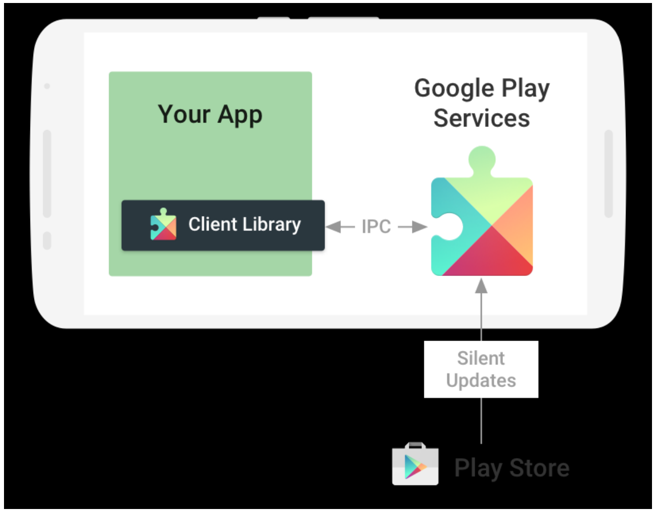
https://www.youtube.com/watch?v=S8sugXgUVEI

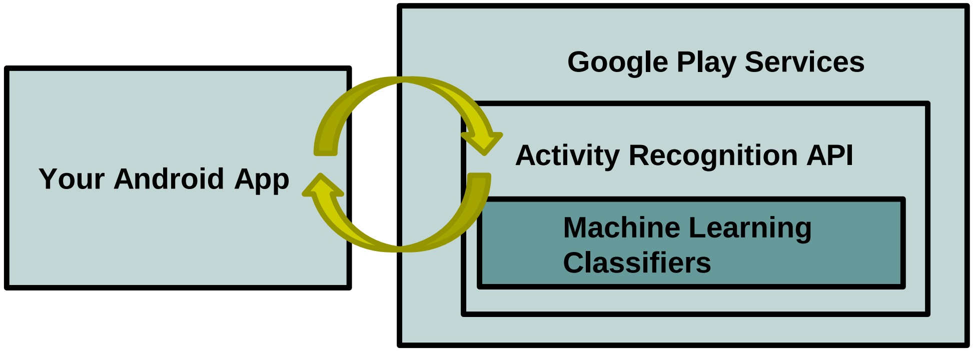
Google Activity Recognition API

- API to detect smartphone user's current activity
- Programmable, can be used by your Android app
- Currently detects 8 states:
  - In vehicle
  - On Bicycle
  - On Foot
  - Running
  - Walking
  - Still
  - Tilting
  - Unknown

Google Activity Recognition API

- Deployed as part of Google Play Services





Your Android App

Google Play Services

Activity Recognition API

Machine Learning Classifiers

Activity Recognition Using AR API

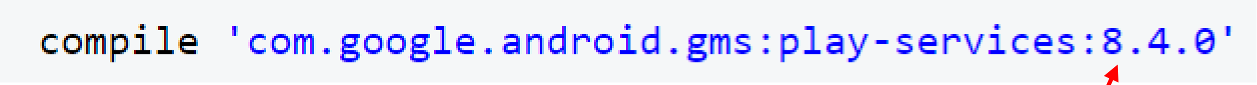
Ref: How to Recognize User Activity with Activity Recognition by Paul Trebilcox-Ruiz on Tutsplus.com tutorials

- 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 (now 11.8) as dependency



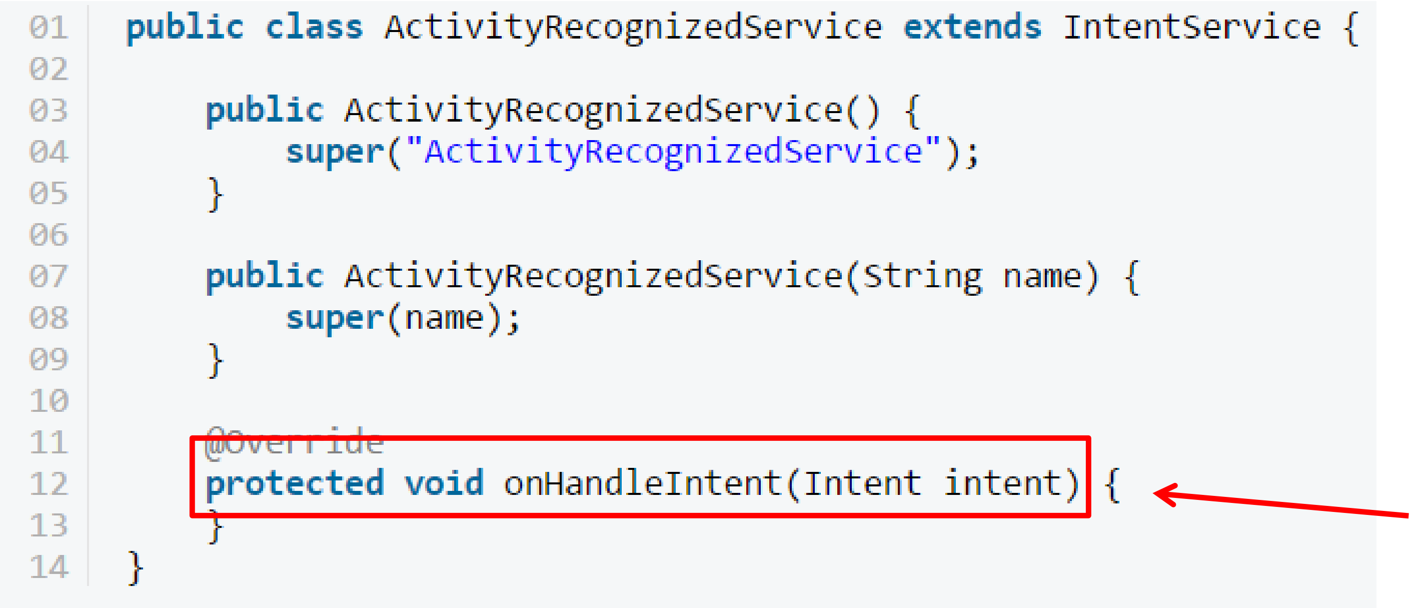
```
compile 'com.google.android.gms:play-services:8.4.0'
```

Now currently Version 11.8.0

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
- 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



```java
public class ActivityRecognizedService extends IntentService {

    public ActivityRecognizedService() {
        super("ActivityRecognizedService");
    }

    public ActivityRecognizedService(String name) {
        super(name);
    }

    @Override
    protected void onHandleIntent(Intent intent) {
    }
}
```

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

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tutsplus.activityrecognition">

    <uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />

    <application
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <service android:name=".ActivityRecognizedService" />
    </application>

</manifest>
```

Requesting Activity Recognition

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

```
public class MainActivity extends AppCompatActivity implements GoogleApiClient.ConnectionCallbacks,
GoogleApiClient.OnConnectionFailedListener {

    public GoogleApiClient mApiClient;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {

    }

    @Override
    public void onConnectionSuspended(int i) {

    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {

    }
}
```

Handle to Google Activity Recognition client

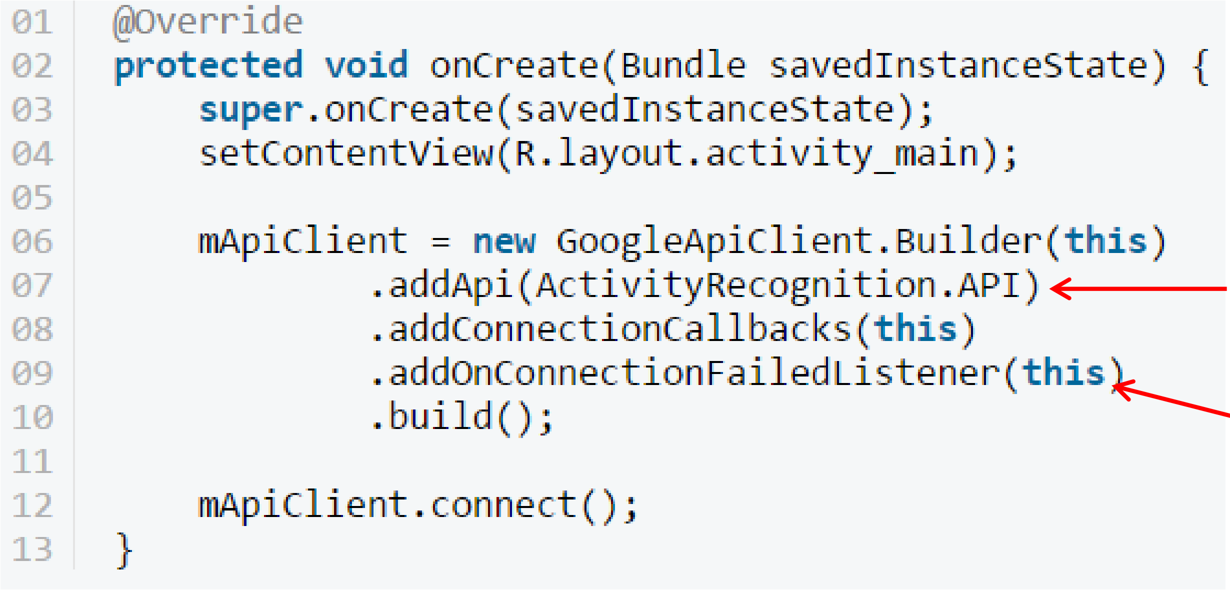
Normal AR call if everything working well

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



```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    mApiClient = new GoogleApiClient.Builder(this)
            .addApi(ActivityRecognition.API)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .build();

    mApiClient.connect();
}
```

Request ActivityRecognition.API

Associate listeners with our instance of GoogleApiClient

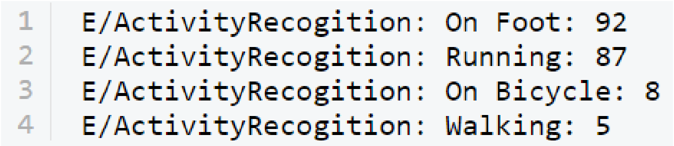
Handling Activity Recognition

- 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



```
E/ActivityRecogition: On Foot: 92
E/ActivityRecogition: Running: 87
E/ActivityRecogition: On Bicycle: 8
E/ActivityRecogition: Walking: 5
```

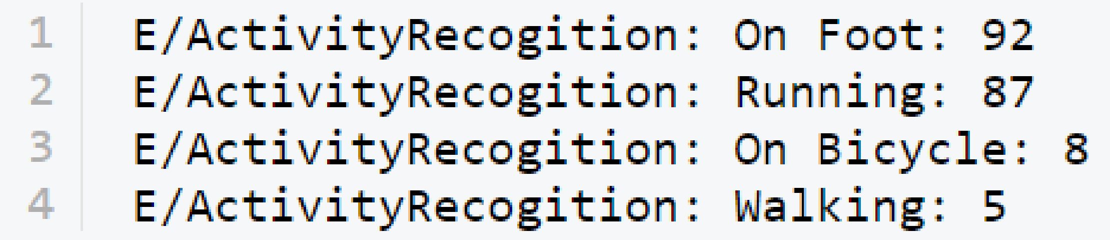
Handling Activity Recognition

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

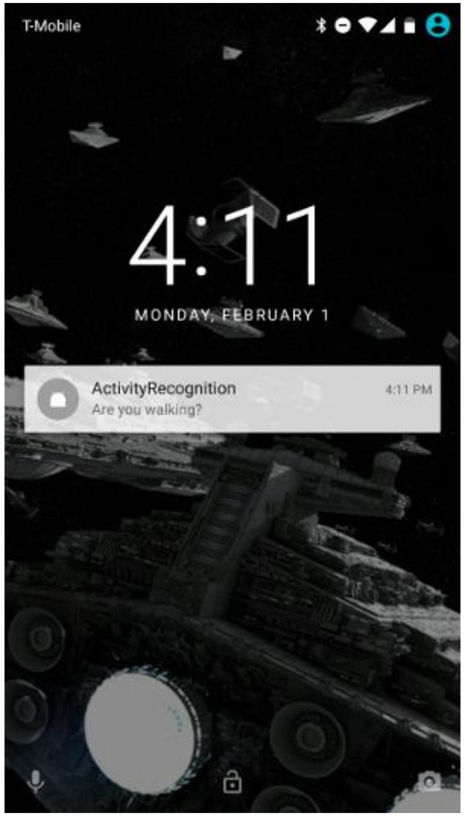
```
            case DetectedActivity.WALKING: {
                Log.e( "ActivityRecogition", "Walking: " + activity.getConfidence() );
                if( activity.getConfidence() >= 75 ) {
                    NotificationCompat.Builder builder = new NotificationCompat.Builder(this);
                    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



```
E/ActivityRecognition: On Foot: 92
E/ActivityRecognition: Running: 87
E/ActivityRecognition: On Bicycle: 8
E/ActivityRecognition: Walking: 5
```



ActivityRecognition
Are you walking?

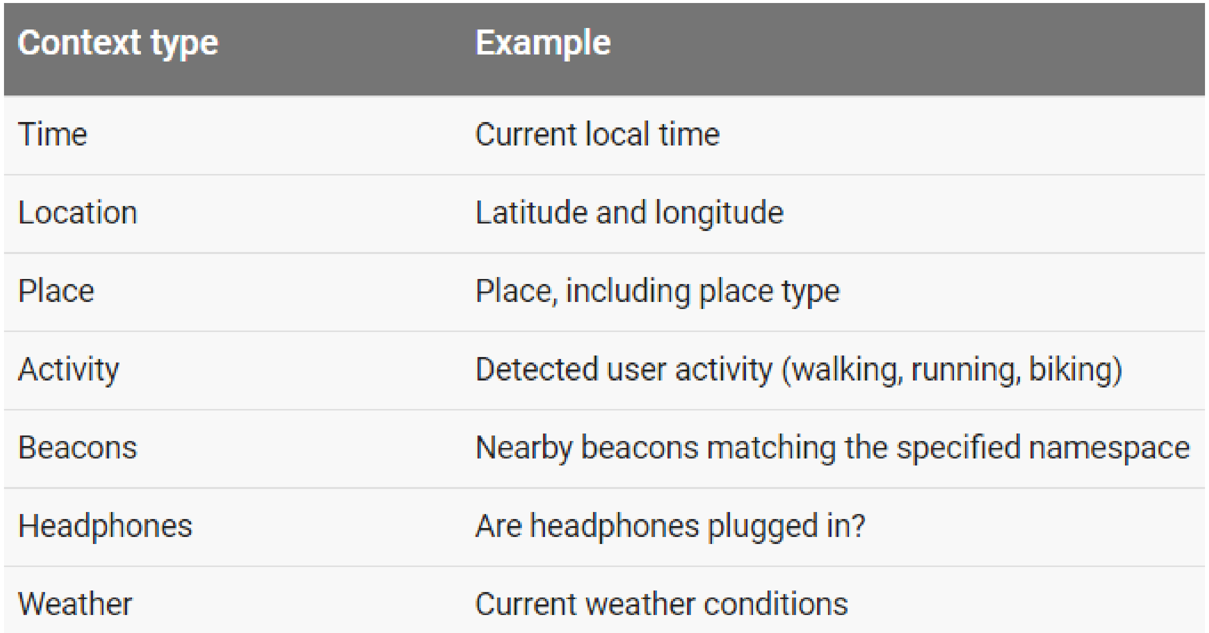
- Full code at: https://github.com/tutsplus/Android-ActivityRecognition

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

Awareness API

- **Snapshot API:**
  - Return cached values (Nearby Places, weather, Activity, etc)
  - System caches values
  - Optimized for battery and power consumption

- **Fences API:**
  - Used to set conditions to trigger events
  - E.g. if(user enters a geoFence & Activity = running) notify my app

- Good tutorials for Awareness API:
  - Google Play Services: Awareness API by Paul Trebilcox-Ruiz

    https://code.tutsplus.com/tutorials/google-play-services-awareness-api--cms-25858
  - Exploring the Awareness API by Joe Birch

    https://medium.com/exploring-android/exploring-the-new-google-awareness-api-bf45f8060bba

Quiz 3

Quiz 3

- Quiz in class next Thursday (before class Oct 11)
- Short answer questions
- Try to focus on understanding, not memorization
- Covers:
  - Lecture slides for lectures 4a,4b,5a,5b
  - 1 code example from book
    - **HFAD examples:** Odometer (Distance Travelled), Ch 13. pg 541
  - All APIs mentioned so far (sensors, Activity Recognition, maps, location sensing, etc)

References

- Android Sensors Overview, http://developer.android.com/guide/topics/sensors/sensors_overview.html
- 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