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:
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 \((\text{sample}_\text{new} < \text{sample}_\text{old})\) when smoothed waveform crosses dynamic threshold
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)
Estimate Distance Traveled
Ref: Deepak Ganesan, Ch 2 Designing a Pedometer and Calorie Counter

- Calculate distance covered based on number of steps taken

\[ \text{Distance} = \text{number of steps} \times \text{distance per step} \] (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

<table>
<thead>
<tr>
<th>Steps per 2 s</th>
<th>Stride (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0~2</td>
<td>Height/5</td>
</tr>
<tr>
<td>2~3</td>
<td>Height/4</td>
</tr>
<tr>
<td>3~4</td>
<td>Height/3</td>
</tr>
<tr>
<td>4~5</td>
<td>Height/2</td>
</tr>
<tr>
<td>5~6</td>
<td>Height/1.2</td>
</tr>
<tr>
<td>6~8</td>
<td>Height</td>
</tr>
<tr>
<td>&gt;=8</td>
<td>1.2 \times \text{Height}</td>
</tr>
</tbody>
</table>
Estimating Calories Burned
Ref: Deepak Ganesan, Ch 2 Designing a Pedometer and Calorie Counter

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

\[ \text{Speed (in m/s)} = \frac{\text{(no. steps per 2 s} \times \text{stride (in meters))}}{2s} \] (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

<table>
<thead>
<tr>
<th>Running Speed (km/h)</th>
<th>Calories Expended (C/kg/h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>25</td>
</tr>
</tbody>
</table>

- Expressed as an equation

\[ \text{Calories (C/kg/h)} = 1.25 \times \text{running speed (km/h)} \] (3)

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

\[ \text{Calories (C/kg/h)} = 1.25 \times \text{speed (m/s)} \times \frac{3600}{1000} = 4.5 \times \text{speed (m/s)} \] (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 classifiers to classify user’s accelerometer signals
Activity Recognition Overview

Gather Accelerometer data

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 classifiers 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 narrow hallways worsens gait freeze

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

Parkinsons disease
Gait freezing

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
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
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 …**

<table>
<thead>
<tr>
<th>Name</th>
<th>Has a pet dog</th>
<th>Has black hair</th>
<th>Likes to play soccer</th>
<th>Has a blue backpack</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Has a brother</th>
<th>Likes to color</th>
<th>Has a summer birthday</th>
<th>Likes chocolate ice cream</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Likes to eat pizza</th>
<th>Can play an instrument</th>
<th>Has a sister</th>
<th>Likes to swim</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Has brown eyes</th>
<th>Is wearing white shoes</th>
<th>Likes the color red</th>
<th>Has a pet cat</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

©2016 Printacraft 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 its 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
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
    
    ```java
    compile 'com.google.android.gms:play-services:8.4.0'
    ```

    Now currently Version 11.8.0
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) {
    }
}
```
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
<?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

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

    public GoogleApiClient mApiClient;  // Handle to Google Activity Recognition client

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

    @Override
    public void onConnected(@Nullable Bundle bundle) {  // Normal AR call if everything working well
    }

    @Override
    public void onConnectionSuspended(int i) {  // Called if sensor (accelerometer) connection fails
    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {  // 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();
```
Handling Activity Recognition

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

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

Sample output:
```
1 E/ActivityRecognition: On Foot: 92
2 E/ActivityRecognition: Running: 87
3 E/ActivityRecognition: On Bicycle: 8
4 E/ActivityRecognition: Walking: 5
```
Handling Activity Recognition

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

```java
    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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>E/ActivityRecognition: On Foot: 92</td>
</tr>
<tr>
<td>2</td>
<td>E/ActivityRecognition: Running: 87</td>
</tr>
<tr>
<td>3</td>
<td>E/ActivityRecognition: On Bicycle: 8</td>
</tr>
<tr>
<td>4</td>
<td>E/ActivityRecognition: Walking: 5</td>
</tr>
</tbody>
</table>

- 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)

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

- 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