

CS 4518 Mobile and Ubiquitous Computing

Lecture 12: Activity Recognition

Emmanuel Agu





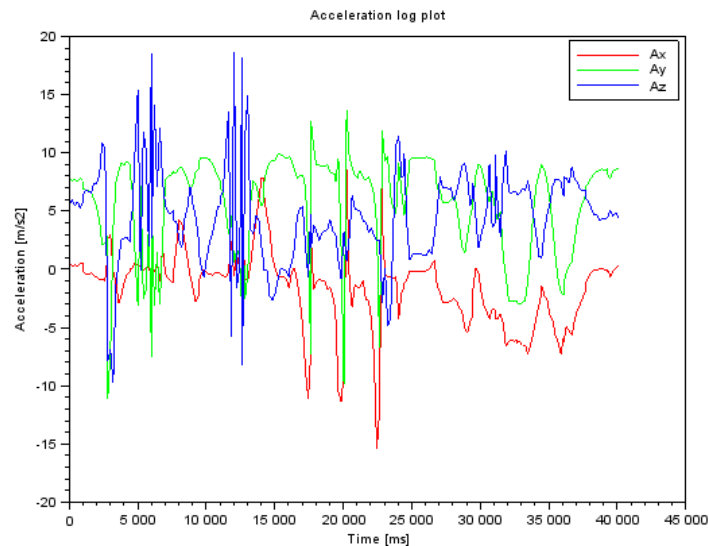
Introduction to Activity Recognition



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

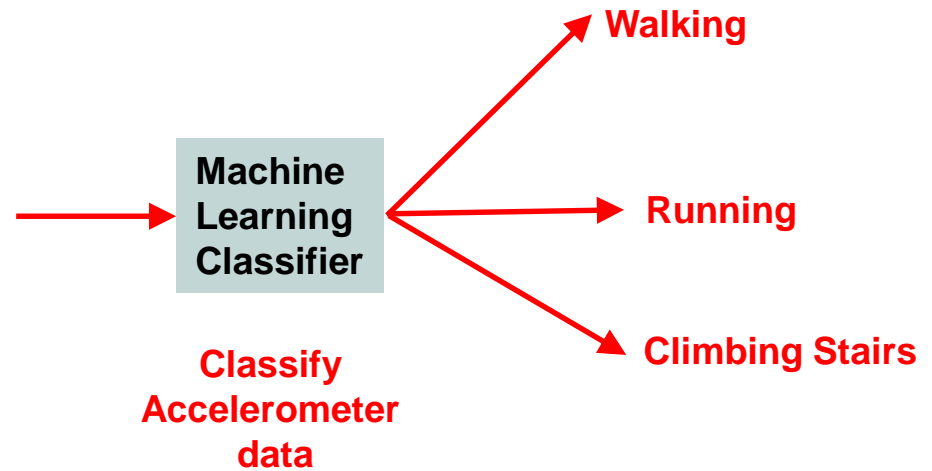
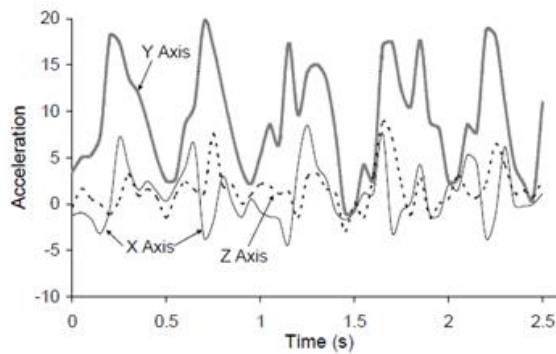


- Typically, use machine learning classifiers to classify user's accelerometer signals

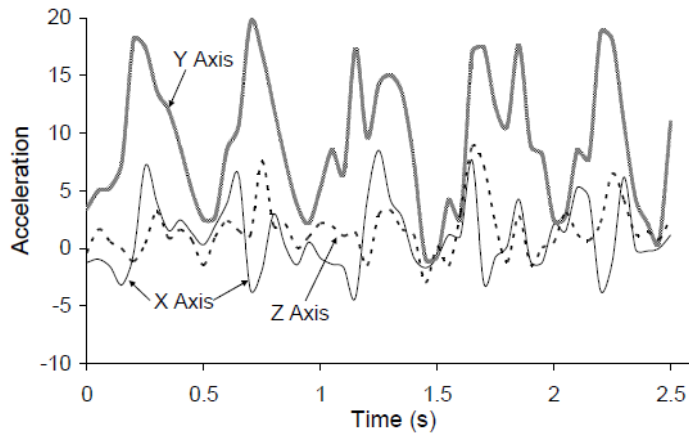
Activity Recognition Overview



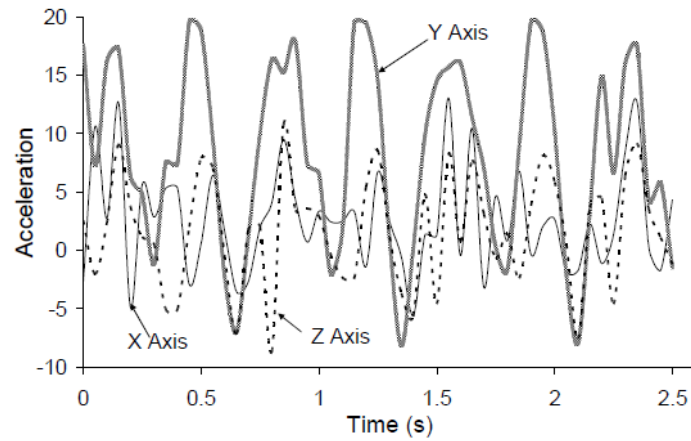
Gather Accelerometer data



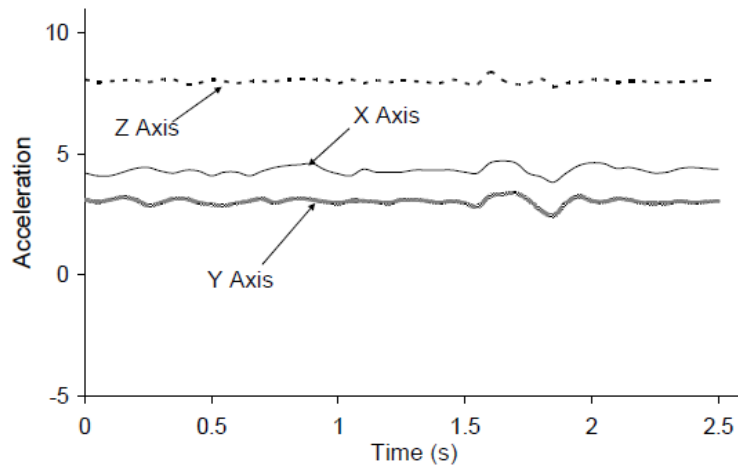
Example Accelerometer Data for Activities



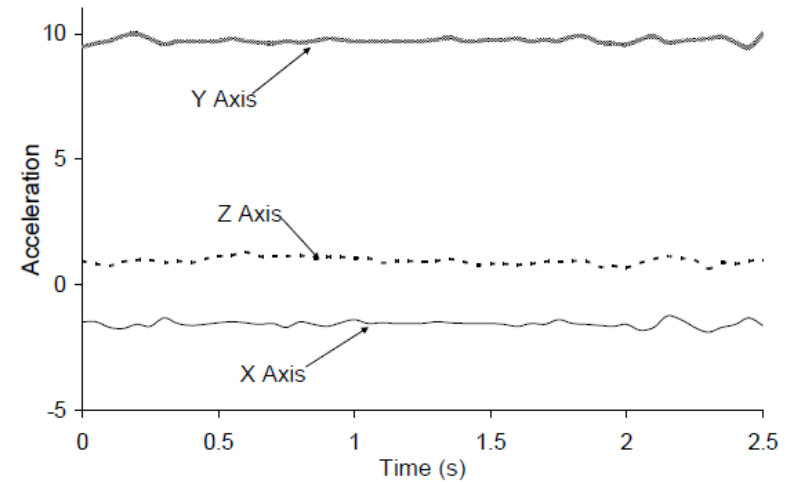
(a) Walking



(b) Jogging

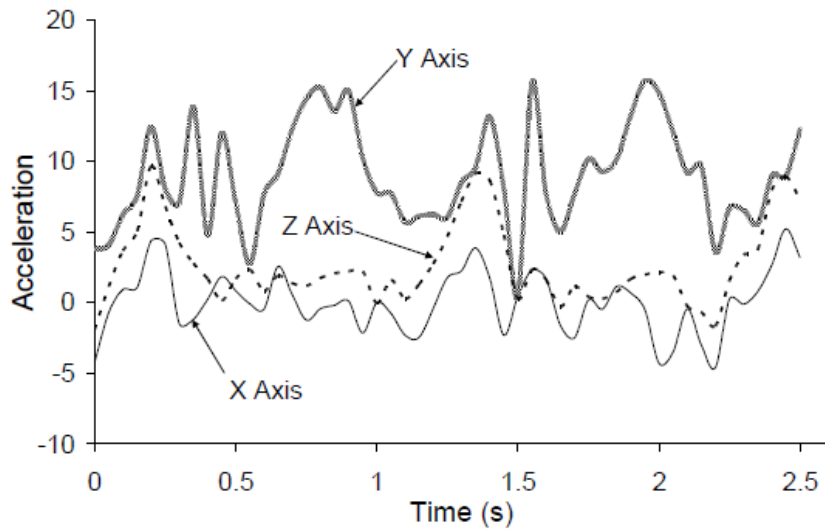


(e) Sitting

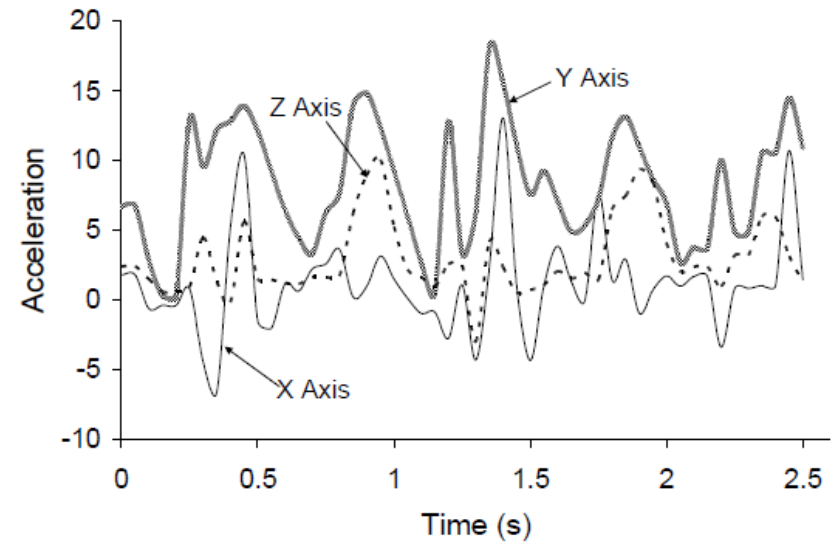


(f) Standing

Example Accelerometer Data for Activities



(c) Ascending Stairs



(d) Descending Stairs



Activity Recognition Using Google API

Activity Recognition



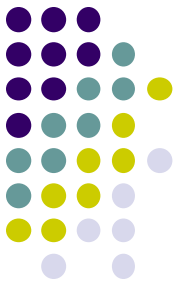
- Activity Recognition? Detect what user is doing?
 - Part of user's context
- Examples: sitting, running, driving, walking
- Why? App can adapt it's behavior based on user behavior
- **E.g.** If user is driving, don't send notifications



<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](https://tutsplus.com) tutorials

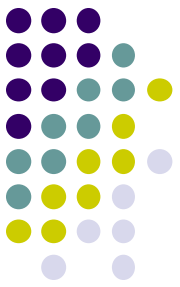


- Example code for this tutorial on gitHub:
<https://github.com/tutsplus/Android-ActivityRecognition>
- Google Activity Recognition can:
 - Recognize user's current activity (Running, walking, in a vehicle or still)
- Project Setup:
 - Create Android Studio project with blank Activity (minimum SDK 14)
 - In **build.gradle** file, define latest Google Play services (was 8.4 last year, now 10.2.7) 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     public GoogleApiClient mApiClient; ← Handle to Google Activity
04                                     Recognition client
05     @Override
06     protected void onCreate(Bundle savedInstanceState) {
07         super.onCreate(savedInstanceState);
08         setContentView(R.layout.activity_main);
09     }
10
11     @Override
12     public void onConnected(@Nullable Bundle bundle) { ← Normal AR call if everything
13                                                         working well
14     }
15
16     @Override
17     public void onConnectionSuspended(int i) { ← Called if sensor (accelerometer)
18                                                         connection fails
19     }
20
21     @Override
22     public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
23         ← Called if Google Play connection fails
24     }
25 }
```



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

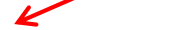


Requesting Activity Recognition

- 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
2 public 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

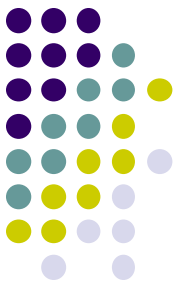
```
1 @Override
2 protected void onHandleIntent(Intent intent) {
3     if(ActivityRecognitionResult.hasResult(intent)) {
4         ActivityRecognitionResult result = ActivityRecognitionResult.extractResult(intent);
5         handleDetectedActivities( result.getProbableActivities() );
6     }
7 }
```

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( "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;
            }
        }
    }
}
```

Switch statement on activity type

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?”

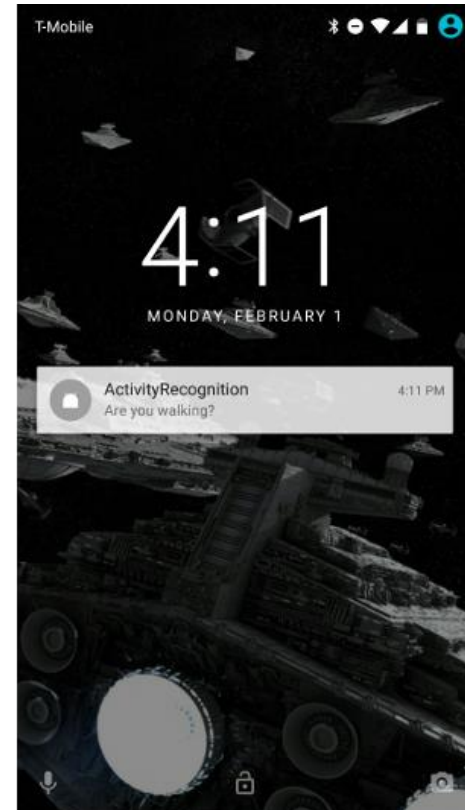
```
case DetectedActivity.WALKING: {
    Log.e( "ActivityRecognition", "Walking: " + activity.getConfidence() );
    if( activity.getConfidence() >= 75 ) {
        NotificationCompat.Builder builder = new NotificationCompat.Builder(this);
        builder.setContentText( "Are you walking?" );
        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( "ActivityRecognition", "Unknown: " + activity.getConfidence() );
    break;
}
}
}
```



Sample Output of Program

- Sample displayed on development console

```
1 E/ActivityRecognition: On Foot: 92
2 E/ActivityRecognition: Running: 87
3 E/ActivityRecognition: On Bicycle: 8
4 E/ActivityRecognition: 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)

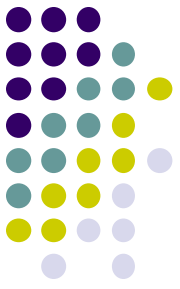
Context type	Example
Time	Current local time
Location	Latitude and longitude
Place	Place, including place type
Activity	Detected user activity (walking, running, biking)
Beacons	Nearby beacons matching the specified namespace
Headphones	Are headphones plugged in?
Weather	Current weather conditions



Quiz 3

Quiz 3

- Quiz in class next Monday (before class Mon, 1/30)
- Short answer questions
- Try to focus on understanding, not memorization
- Covers:
 - Lecture slides for lectures 9-12
 - 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