Location-Aware Computing

- **Definition**: Location-aware applications generate outputs/behaviors that depend on a user’s location.

- **Examples**:
  - Map of user’s “current location”
  - Print to “closest” printer
  - Apps that find user’s friends “closeby”
  - Reviews of “closeby” restaurants

- Apps above require first determining user’s location.
Determining User Location on Smartphones
Location Tracking on Smartphones

- **Outdoors**: Uses GPS (More accurate)
- **Indoors**: WiFi or cell tower signals (Location fingerprinting, less accurate)
Global Positioning System (GPS)

- 27 satellites orbiting earth
- **20,000 km above earth** (Medium earth orbit)
- 6 orbital planes with 4 satellites each
- 4 satellites visible from any spot on earth
- Location of any location on earth specified as `<longitude,latitude>`
- E.g. Worcester MA has **Latitude:** 42.2625, **Longitude:** -71.8027778
GPS User Segment

- **Triangulation**: GPS receiver calculates user’s position by comparing roundtrip delay of signals to multiple satellites at known positions.

- Accuracy within 5 - 10 meters (16-32 feet)

http://adamswalk.com/gpx-2/
Determining User Location

- GPS reasonably accurate but
  - Requires line-of-sight between satellite and car receiver
  - Only works OUTDOORS (signals don’t penetrate buildings)
  - **Lag/delay** in acquiring satellites (~270 msec) or re-acquiring if lost
  - Drains battery power

- **Alternative:** Use Wi-Fi location sensing indoors
WiFi Location Fingerprinting

- **Key insight:** At each (X,Y) location, WiFi APs observed + their signal strengths, is unique

- **WiFi Location fingerprinting:** Infer device’s location based on combination of Wi-Fi access points seen + Signal Strengths
Location Estimation using Wi-Fi Fingerprinting

Google builds and stores this database (APs + Signal Strength) at each X,Y location)

<table>
<thead>
<tr>
<th>Location</th>
<th>Signal Strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>X Y AP1 AP2 AP3 AP4</td>
<td></td>
</tr>
<tr>
<td>80 145 32 28 12 8</td>
<td></td>
</tr>
<tr>
<td>40 145 36 20 10 6</td>
<td></td>
</tr>
<tr>
<td>220 355 - 25 36 44</td>
<td></td>
</tr>
<tr>
<td>260 355 4 21 39 42</td>
<td></td>
</tr>
<tr>
<td>350 210 16 - 28 36</td>
<td></td>
</tr>
<tr>
<td>380 145 22 12 - 44</td>
<td></td>
</tr>
</tbody>
</table>

Inference Algorithms
- Min. Threshold
- Euclidean Dist.
- Joint Probability
- Bayesian Filters
How to Build table of APs observed at (X,Y) Locations?

- Devices (e.g. smartphone) with GPS and WiFi turned on simultaneously build table
- Send data to third party repositories (e.g. Wigle.net) or Google
- Also called **war driving**
- Can record cell tower signal strength instead of AP

---

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>SIGNAL STRENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>80</td>
<td>145</td>
</tr>
<tr>
<td>40</td>
<td>145</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>220</td>
<td>355</td>
</tr>
<tr>
<td>260</td>
<td>355</td>
</tr>
</tbody>
</table>
Location Sensing in Android Apps
Google Location APIs


- Android now has 2 location APIs (older vs newer)
- Newer location API is now part of Google Play Services
- Older Android framework location APIs (android.location)
  - Used by most books, online sources. We will use that

**LocationManager:**
- Android module receives location updates from GPS, WiFi, etc
- App registers/requests location updates from LocationManager

```java
Your app
requestLocationUpdates(LocationListener)
```

- `onStatusChanged`
- `onProviderEnabled`
- `onProviderDisabled`
Requesting Location Updates

Your app

LocationManager

onStatusChanged

onProviderEnabled

onProviderDisabled

// Acquire a reference to the system Location Manager
LocationManager locationManager = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);

// Define a listener that responds to location updates
LocationListener locationListener = new LocationListener() {
    public void onLocationChanged(Location location) {
        // Called when a new location is found by the network location provider.
        makeUseOfNewLocation(location);
    }

    public void onStatusChanged(String provider, int status, Bundle extras) {}

    public void onProviderEnabled(String provider) {}

    public void onProviderDisabled(String provider) {};
};

// Register the listener with the Location Manager to receive location updates
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, locationListener);
Requesting User Permissions

- Need smartphone owner’s permission to use their GPS

```xml
<manifest ...
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    ...
    <!-- Needed only if your app targets Android 5.0 (API level 21) or higher. -->
    <uses-feature android:name="android.hardware.location.gps" />
    ...
</manifest>
```

- ACCESS_FINE_LOCATION: GPS
- ACCESS_COARSE_LOCATION: WiFi or cell towers
Getting Cached Copy of Location (Fast)

- Getting current location may take a while
- Can choose to use location cached (possibly stale) from Location Manager

```java
String locationProvider = LocationManager.NETWORK_PROVIDER;
// Or use LocationManager.GPS_PROVIDER

Location lastKnownLocation = locationManager.getLastKnownLocation(locationProvider);
```
Stopping Listening for Location Updates

- Location updates consume battery power
- Stop listening for location updates whenever you no longer need

```java
// Remove the listener you previously added
locationManager.removeUpdates(locationListener);
```
Distance Travelled Updates using Services
Example from Head First Android
Example: Odometer (Distance Travelled) updates as a Services
(Ref: Head First Android pg 541)

- **Services:** long running background processes, no UI

- May want background service (a module in our app) to continuously retrieve location updates from LocationManager, forward updates to our Activity

- Ref: Head First Android pg 541
  - Example of using a Service
  - Nice Example app using Odometer Service
  - Tracks distance travelled
  - Gets, displays distance travelled every 10 secs
Example: Odometer (Distance Travelled) updates as a Services
(Ref: Head First Android pg 541)

- Example odometer app that tracks distance travelled
- `getMiles()`, displays distance travelled every 10 seconds

Study this example!!!
Location Representation
Semantic Location

- GPS represents location as \(<\text{longitude},\text{latitude}>\)
- **Semantic location** is better for reasoning about locations
- E.g. Street address (140 Park Avenue, Worcester, MA) or (building, floor, room)
- **Android supports:**
  - **Geocoding:** Convert addresses into longitude/latitude coordinates
  - **Reverse geocoding:** convert longitude/latitude coordinates into human readable address

  ![Location coordinates and details](image)

- **Android Geocoding API:** access to **geocoding** and **reverse geocoding** services using HTTP requests
Google Places API Overview

- **Access high-quality photos** of a place
- Users can also add place information to the database
  - E.g. business owners can add their business as a place in Places database
  - Other apps can then retrieve info after moderation

**On-device caching:** Can cache places data locally on device to avoid roundtrip delays on future requests
Google Places

- **Place**: physical space that has a name (e.g. local businesses, points of interest, geographic locations)
  - E.g Logan airport, place type is **airport**

- **API**: Provides Contextual information about places near device.

- **E.g:** name of place, address, geographical location, place ID, phone number, place type, website URL, etc.

- Compliments geographic-based services offered by Android location services
Sample Place Types

accounting
airport
amusement_park
aquarium
art_gallery
atm
bakery
bank
bar
beauty_salon
bicycle_store
book_store
bowling_alley
bus_station
cafe
campground
car_dealer
car_rental
car_repair
car_wash
hospital
insurance_agency
jewelry_store
laundry
lawyer
library
liquor_store
local_government_office
locksmith
lodging
meal_delivery
meal_takeaway
mosque
movie_rental
movie_theater
moving_company
museum
night_club
painter
park
city_hall
clothing_store
convenience_store
courthouse
dentist
department_store
doctor
electrician
electronics_store
embassy
establishment (deprecated)
finance (deprecated)
fire_station
florist
food (deprecated)
funeral_home
furniture_store
gas_station
general_contractor (deprecated)
grocery_or_supermarket
gym
hair_care
hardware_store
health (deprecated)
hindu_temple
home_goods_store
physiotherapist
place_of_worship (deprecated)
plumber
police
post_office
real_estate_agency
restaurant
roofing_contractor
erv_park
school
shoe_store
shopping_mall
spa
stadium
storage
store
subway_station
synagogue
taxi_stand
train_station
transit_station
travel_agency
university
veterinary-care
zoo
Google Places API Overview

- **Use Place picker UI:** allows users select place from “possible place” on a map

- **Get current place:** place where device is last known to be located
  - Returns **list** of likely places + likelihood device is in that place
Google Places API Overview

**Autocomplete**: queries the location database as users type, suggests nearby places matching letters typed in.
Learning Google Places API

- Official Google Places website is “decent”, up to date:
  - https://developers.google.com/places/

- Two great references:
  a) Getting started with Google Places API
     https://developers.google.com/places/android-api/start
  
  b) Tutorial by Paul Trebilcox-Ruiz may be more readable:
     - http://code.tutsplus.com/articles/google-play-services-using-the-places-api--cms-23715
Other Useful Google Maps/Location APIs
GeoFencing
https://developer.android.com/training/location/geofencing.html

- **Geofence**: Sends alerts when user is within a certain radius to a location of interest

- Can be configured to send:
  - **ENTER** event when user enters circle
  - **EXIT** event when user exits circle

- Can also specify a duration or **DWELL** user must be in circle before triggering event
GeoFencing

https://developer.android.com/training/location/geofencing.html

- **Great reference:**
  - How to work with GeoFences on Android by Tin Megali
    https://code.tutsplus.com/tutorials/how-to-work-with-geofences-on-android--cms-26639
Other Maps/Useful Location APIs

- **Maps Directions API**: calculates directions between locations (walking, driving) as well as public transport directions
- **Distance Matrix API**: Calculate travel time and distance for multiple destinations
- **Elevation API**: Query locations on earth for elevation information, calculate elevation changes along routes
Other Useful Maps/Location APIs

- **Rocks API:**
  - snaps set of GPS coordinates to road user was likely travelling on (best fit)
  - Returns posted speed limits for any road segment (premium plan)

- **Time Zone API:** request time zone for location on earth
GPS Clustering & Analytics
Determining Points of Interest from GPS Location Sequences

- **Points of Interest**: Places where a person spends lots of time (e.g. home, work, café, etc)
- **Given a sequence GPS <longitude, latitude> points**, how to infer points of interest
- **General steps**:  
  - Pre-process sequence of GPS points (remove outliers, etc)  
  - Cluster points  
  - Convert to semantic location

<table>
<thead>
<tr>
<th>LATITUDE</th>
<th>LONGITUDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>35.33032098</td>
<td>80.42152478</td>
</tr>
<tr>
<td>35.29244028</td>
<td>80.42382271</td>
</tr>
<tr>
<td>35.33021993</td>
<td>80.45339956</td>
</tr>
<tr>
<td>35.35529007</td>
<td>80.45222096</td>
</tr>
</tbody>
</table>
Step 1: Pre-Processing GPS Points
(Remove Noise and Outliers)

- **Remove low density points (few neighbors):**
  - i.e. places where little time was spent
  - E.g. radius of 20 meters, keep only clusters with at least 50 points
  - If GPS coordinates retrieved every minute, only considering places where you spent at least 50 minutes

- **Remove points with movement:**
  - GPS returns speed as well as <longitude, latitude> coordinates
  - If speed user is moving, discard that GPS point

- **Reduce data for stationary locations:**
  - When user is stationary at same location for long time, too many points generated (e.g. sitting at a chair)
  - Remove some points to speed up processing
Step 2: Cluster GPS Points

- **Cluster Analysis**: Group points

- Two main clustering approaches
  - K-means clustering
  - DBSCAN
K-Means Clustering

- Each cluster has a center point (centroid)
- Each point associated to cluster with closest centroid
- Number of clusters, $K$, must be specified
- Algorithm:

```plaintext
1: Select $K$ points as the initial centroids.
2: repeat
3: Form $K$ clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: until The centroids don’t change
```
DBSCAN Clustering

- Density-based clustering

- **Density**: Number of points within specified radius (Eps)

- **Core points**: has > minPoints density

- **Border point**: has < minPoints density but within neighborhood of core point

- **Noise point**: not core point or border point
DBSCAN Algorithm

- Eliminate noise points
- Cluster remaining points

```python
current_cluster_label ← 1
for all core points do
    if the core point has no cluster label then
        current_cluster_label ← current_cluster_label + 1
        Label the current core point with cluster label current_cluster_label
    end if
    for all points in the $Eps$-neighborhood, except $i^{th}$ the point itself do
        if the point does not have a cluster label then
            Label the point with cluster label current_cluster_label
        end if
    end for
end for
```
Converting Clusters to Semantic Locations

- Can simply call reverse geocoding or Google Places on the centroid of the clusters
Visualizing Points of Interests visited
Visualizing Points of Interest

• Option 1:

  • Show a point for each location you visited?
Visualizing Points of Interest

- Option 2:
  - Show a cluster for significant locations.

Credit: Deepak Ganesan
Visualizing Points of Interest

• Option 3:

• Connect the clusters with lines
Visualizing Points of Interest

• Option 4
  • Show “semantic locations” instead of co-ordinates
  • Use size of circle to represent duration of stay
Visualizing Points of Interest

• Option 5

• Show semantic locations with time-of-day encoded in line opacity/saturation.
Location-Aware Apps from CS 4518
Location-Aware Final Projects from CS 4518 (Undergraduate offering)

- **Ground rules:**
  - Apps must use mobile, location or sensors
  - Try to solve problems of benefit to WPI community

- More than half of apps used location.

- **Give me some space:** Bianchi, Chow, Martinez ’16
  - Find available study spaces on campus during exam week
  - Set up geoFences at study locations, count users in/out
Location-Aware Ideas from Previous Offerings

- **HomeSafe**: Nickerson, Feeley, Faust ’16
  - Safety app
  - Automatically sends message to users’ subscribers when they get home safely
Some Interesting Location-Aware Apps
MileIQ

- **The Problem:** Mileage tracking is useful but a burden.
  - IRS deductions on taxes
  - Some companies reimburse employees for mileage,
- Passively, automatically tracks business mileage, IRS compliant
- Swipe right after drive to indicate it was a business trip
- Project idea? Implement some of this functionality

- How Android modules? For what?
- What stats to decide if this is tackling important problem?
Trigger

- Use geofences, NFC, bluetooth, WiFi connections, etc to set auto-behaviors
  - Battery low -> turn off bluetooth + auto sync
  - Silence phone every morning when you get to work
  - Turn off mobile data when you connect to your home WiFi
  - Silence phone and set alarm once I get into bed
  - Use geofence for automatic foursquare checkin
  - Launch maps when you connect to your car’s bluetooth network
- Project idea? Implement subset of these features

- What triggers would be useful for a WPI student?
References

- John Corpuz, 10 Best Location Aware Apps
- Liane Cassavoy, 21 Awesome GPS and Location-Aware Apps for Android,
- Head First Android
- Android Nerd Ranch, 2\textsuperscript{nd} 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