

University of Southern California
School of Engineering
USC Viterbi

Introduction to GPS

Slides based on materials from

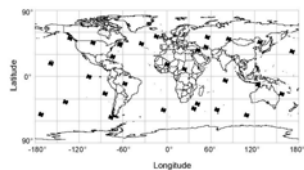
[1] TRIMBLE GPS Tutorial, online at <http://www.trimble.com/gps/index.shtml>

[2] Jean-Marie Zogg, *GPS Basics*, online at <http://www.u-blox.cn/customersupport/docs/GPS-X-02007.pdf>

[3] Wikipedia entry on Global Positioning System

GPS

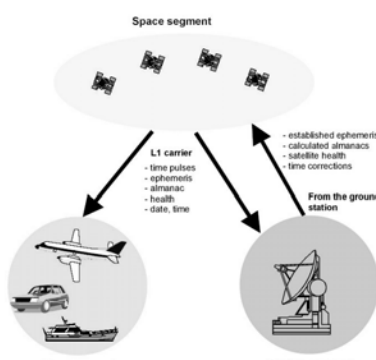
- Based on radio systems from 1940s, started in late 1970s.
- Gives longitude, latitude, height (with meter-scale accuracy), time (within 60 ns accuracy)
- Based on a signals from a system of ~30 MEO satellites (~20000 km altitude)



GPS Segments

Extremely noisy Link: -15dB SNR!

Data rate: 50 bits per second



Space segment

- established ephemeris
- calculated almanacs
- satellite health
- time corrections

From the ground station

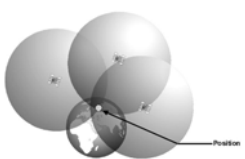
- L1 carrier
- time pulses
- ephemeris
- almanac
- health
- date, time

User segment

Control segment

Basic Idea

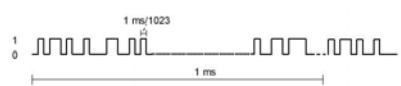
- Estimating distance from 3 satellites can fix location



- Distance estimation is based on timing, which itself has errors, so a 4th satellite is needed to solve for four unknowns (x,y,z, and t)

PN Sequences

- Each satellite has a unique signature



- Used to identify satellite and obtain time-difference estimate

Transmitter

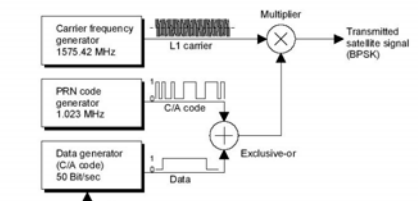
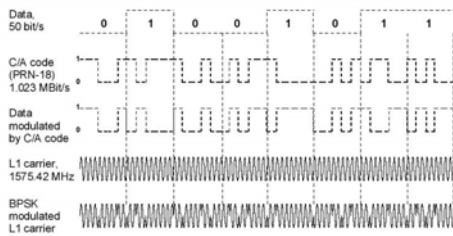
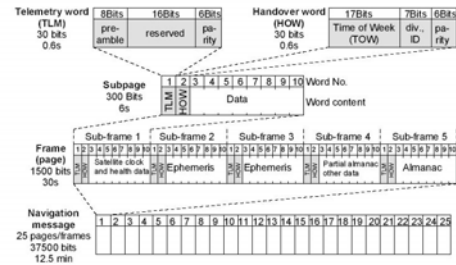


Figure 13: Simplified satellite block diagram

Signal Modulation



Navigation Message Format



Accurate Timing

- Satellites have atomic clocks
- Receivers do not. Hence the use of an additional satellite to correct for timing errors.

Satellite Position

- Need to know where satellites are located
- Because of predetermined orbits, their locations are obtained through an almanac program on the receivers
- Slight errors in the precomputed orbits (e.g., due to the moon's gravitational pull) – ephemeris measurements, which are measured from the ground, then repeated back from the satellites

Error Sources

Summary of GPS Error Sources

Typical Error in Meters (per satellites)	Standard GPS
Satellite Clocks	1.5
Orbit Errors	2.5
Ionosphere	5.0
Troposphere	0.5
Receiver Noise	0.3
Multipath	0.6

Error Sources

Summary of GPS Error Sources

Typical Error in Meters (per satellites)	Standard GPS	Differential GPS
Satellite Clocks	1.5	0
Orbit Errors	2.5	0
Ionosphere	5.0	0.4
Troposphere	0.5	0.2
Receiver Noise	0.3	0.3
Multipath	0.6	0.6

Differential GPS

- Have a reference receiver with known location
- The reference station works backwards on GPS signals.
- "Instead of using timing signals to calculate its position, it uses its known position to calculate timing. It figures out what the travel time of the GPS signals should be, and compares it with what they actually are. The difference is an 'error correction' factor." [1]
- The reference then transmits error information to mobile receivers over a radio link.



Wide Area Augmentation System (WAAS)

- An FAA system to provide the differential corrections
- Ground stations to measure errors, geostationary satellites to broadcast these
- Typically provides 1m accuracy



A-GPS

- Uses cellular network to obtain error correction from a server
- Also provides faster fix




Relativity matters!

- Special Relativity: satellite clocks slow by 7us per day due to the time dilation effect.
- General Relativity: clocks close to massive object are slower – so earth-bound clocks are slow by 45 us per day.
- Without correction, would lose 10 kilometers each day!
- Satellite clocks are therefore pre-corrected by 38 us per day.

Source: <http://www.astronomy.ohio-state.edu/~pogge/Ast162/Unit5/gps.html>






University of Southern California
USC Viterbi
School of Engineering

Location API: Introduction to JSR 179

Murali Annavaram & Bhaskar Krishnamachari
Ming Hsieh Department of Electrical Engineering
USC

Lecture notes based in part on slides created by Vidya Setlur,
Thanks also to Mikko H. Lipasti for course suggestions.



Why Location?

- Mobile devices go wherever we go – whoever thought laptops are mobile did not think about boot latency
- Mobile devices are increasingly capable of providing location information
 - Use integrated GPS chip (N95)
 - Use Bluetooth based GPS communicate with cell phone
 - Use cell towers to get approximate location (Google mobile maps)
- LBS is expected to be the fastest growing service sector
 - Most interested application people are interested in subscribing to
 - advising users of current traffic conditions, supplying routing information, helping them find nearby restaurants, and many more
 - 300 million phones with GPS in less than 2 years



Early Challenges to LBS

- Location access challenges
 - Different methods, different platforms and accuracies
 - Device based and network based solutions
- Mapping challenges
 - Map providers used several coordinate systems and APIs
- Privacy of location data
 - People are sensitive to revealing location data



LBS Support in J2ME : JSR 179

- JSR 179 standardizes location retrieval in Java ME
 - Supports synchronous and asynchronous (listener based) retrieval of location information
 - Supports tagging the location using built-in Landmarks database
- More features supported in JSR 293
 - Geocoding, reverse geocoding, navigation, mapping



Basic Steps in LBS

- Select the criteria for LBS provider
 - Horizontal/vertical accuracy, preferred response time, cost allowed, altitude required, power consumption, ...
 - E.g. Within 50 meter accuracy, should be free
 - Criteria cr= new Criteria();
 - cr.setHorizontalAccuracy(50);
 - cr.setCostAllowed(false);
- Get an instance of location provider with selected criteria
 - Mylp = LocationProvider.getInstance(cr)



Approach1: Get Location Synchronously

- Get location from the LocationProvider instance synchronously
 - Location = Mylp.getLocation(timeout_interval)
 - The method is blocking call and waits for the providers to give us location data or until the timeout interval expires
 - Throws locationexception if failed
 - Also note, that the location returned does not necessarily satisfy the criteria we used
 - Criteria is only used during the locationprovider selection but the provider may not really honor it's commitment
- Get the coordinates from the location
 - Coordinate = location.getQualifiedCoordinates()
- Coordinates are lat, long values
 - coordinate.getLatitude(), coordinate.getLongitude()



Approach 2: Get Location Asynchronously

- Attach a location listener to the location provider
 - mylp.setLocationListener(listenerclass, interval, timeout, age);
 - The listener will be called with updated location at the defined interval
 - The timeout is used to determine the maximum waiting time after the interval time to get the location (an invalid location update is then made)
 - If the interval is 1 minute and timeout is 20 seconds, then update must be delivered at most 80 seconds after previous update
 - Age defines how often the underlying hardware needs to compute the location since the hardware may reuse the old location
- Only one listener can be attached per each provider



LocationListener

- Location information is provided by call back to LocationUpdated
 - locationUpdated(LocationProvider provider, Location location)
 - Called by the LocationProvider to which this listener is registered
- Listener can then process the location information to get the location latitude and longitude



Improving Power Efficiency of GPS

- Getlocation method is power hungry
- If you need only rough location you can use `getLastKnownLocation()` method
 - Provides only the last known location
 - Implementation specific and hence may in turn call `getLocation()` anyway



Storing Locations as Landmarks

- JSR 179 also provides a method for associating a location with a name, called Landmarks
 - `Landmark(name, description, coordinates, address)`
- All landmarks are stored in a device specific landmark database called `LandmarkStore`
- Typical steps to add a landmark to database:
 - `Lms = createLandmarkStore(name) or getInstance(name) or listLandmarkStores()`
 - `Lms.addcategory(categoryname) → Sights, Hotels,...`
 - `Lms.addLandMark(landmark, category)`
- Retrive landmarks
 - `Lms.getlandmarks(category, name) → both parameters can be null for wildcard matching`



Advanced LBS : JSR 293 Geocoding

- Geocoding service provider
 - Geocoding: Convert a street address to lat/long coordinates
 - Reversegeocoding: Convert a lat/long to street address
- Example Code

```
// obtaining a service first
GeocodingServiceProvider service =
    (GeocodingServiceProvider)ProviderManager.connectToServiceProv
    der(null,
    ProviderManager.GEOCODING);
Coordinates myCoord = new Coordinates(lat, long, altitude);
results = service.reverseGeocode(myCoord, false);
// pick the first result
myLandmark = (Landmark) results.nextElement();
// Use the landmark
AddressInfo myAddress = myLandmark.getAddressInfo();
```



JSR 293 Mapping Services

- Map service provider
 - Show a map with items of interest (landmarks, route...)
 - Show a map and allow user to select map items
- Example

```
MapServiceProvider mapProvider =
    (MapServiceProvider)ProviderManager.connectToService
    Provider(null, ProviderManager.MAP);
Coordinates myCoord = new Coordinates(lat, long, alt);
// request map service provider to show a map
mapProvider.displayMap(null, null, myCoord, null,
    MapContainer.MAP_TYPE_REGULAR, 0, -1, false, false,
    this);
```



LBS Summary

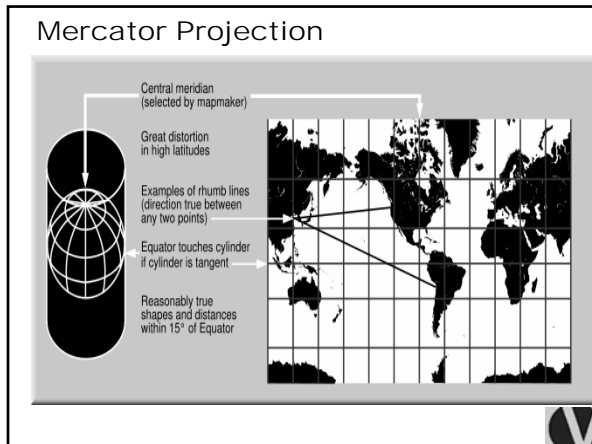
- GPS information is critical component of LBS infrastructure
- JSR 179 and JSR 293 provide the necessary API to access GPS information
- But to be truly useful the location data needs to be processed by a backend server to provide user specific information
 - Most LBS applications have both the client and server component



Mapping Basics: Projection

- While lat/longs give precise location information on globe you need to transform lat/longs to a location on a 2D MAP
- Projection means any mathematical transformation of the globe onto some other surface
- Our challenge is to represent a round earth to a flat surface
 - Several map projection methods exist for this purpose
- We will use Mercator Projection in this class
 - Google Map tile server and our class map tile server use this projection





Mercator Accuracy

- The Mercator is "conformal" map projection
- It shows shapes pretty much the way they appear on the globe
 - But you cannot show both shape and size accurately at the same time
- Mercator projection creates increasing distortions of size as you move away from the equator
- Greenland appears roughly the same size as Africa
 - But Africa is 14X the size of Greenland
- We use Mercator because it is well-suited as an interactive world map that can be panned and zoomed seamlessly

Map Tile Server

- Google converts lat/longs to X,Y coordinates
- Lat/longs converted into 25 bit integers (code for conversion provided to the class later)
- Supports up to 17 zoom levels
- Each zoom above covers 4X the size of previous zoom level
- The world starts as a single square tile with X=0, Y=0, Z=17

Tile Server Contd.

- Each tile is then split into 4 tiles X=00/01/10/11, Y=00/01/10/11 and each covers 1/4 the area at Zoom 16

Tile Server Contd.

- The process of zooming in continues recursively and at each step we add one more bit
 - X=000/001/010/011/100/101/110/111
 - Y=000/001/010/011/100/101/110/111
 - Z=15
- Google map tile server stops at Z=17. Each tile is 25 meters X 25 meters
- <http://mt0.google.com/mt?&n=y=198&zoom=8>

Representing Location as Quadrants

- Location is encoded in a recursive quadrant representation.
- Instead of QRST we use X, Y, Z encoding.
- Z represents that the world is divided into 2^ZX2^Zsquares.
- X and Y are the offset from the top left corner of the world.

Q	RQ				
	RT	<table border="1"> <tr> <td>RSQ</td> <td></td> </tr> <tr> <td>RST</td> <td></td> </tr> </table>	RSQ		RST
RSQ					
RST					
T	S				

Assignment#5

- Use JSR 179 API to programmatically get GPS location
- Convert the location to X,Y coordinates each 25 bits wide
- Access a tile server with X,Y values to get the necessary tile
- Show the map tile on your canvas
- Then use any simple icon to point your position on the map
- Use the keys to zoom in and out one level at a time and continuously update the map



Assignment#5 Contd

- Then combine your previous knowledge from assignment#3 on MMAPI that captures photos programmatically
- Convert the photo into a small icon (say 16X16 pixel)
- Tag the photo with the GPS coordinates
- Finally display the photo icon on the map using the GPS location tags

