Course summary

Björn Bebensee

Mobile and Ubiquitous Computing

June 10, 2020

Contents

1 Human Behavior and Context Sensing

1. Sense real-world situations and human behavior
2. Extract and infer useful insights and Knowledge
3. Provide what people need right on time & place

Mobile devices have many sensors that we can utilize to sense the context (smartphone-embedded sensors, wearable sensors, space/object-embedded sensors). We can use these sensors to classify and sense the context or behavior (see Figure 1).

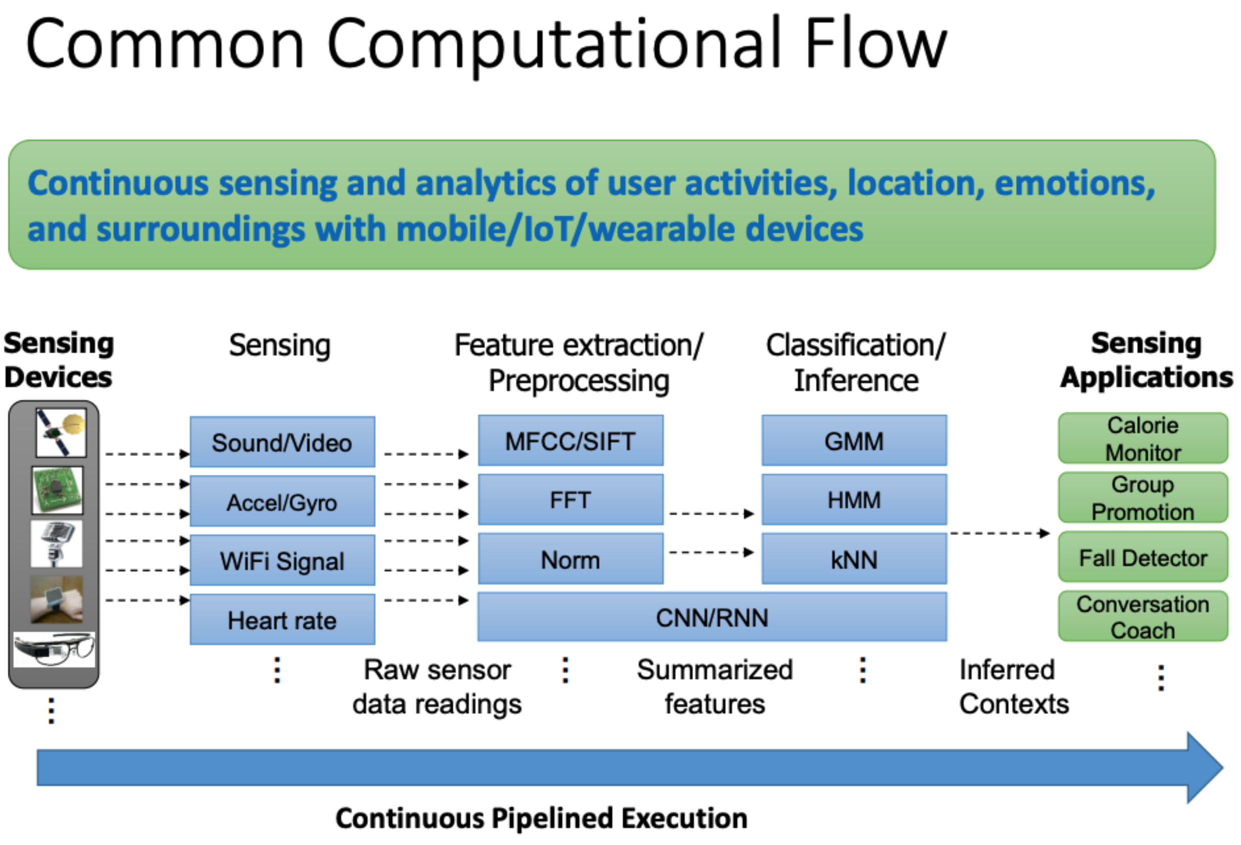


Figure 1: Typical computational flow

1.1 Challenges

1.1.1 Challenge 1: Inference Accuracy

It is however extremely difficult to achieve $> 90\%$ accuracy as errors can be caused in multiple layers and add up: 1) uncalibrated readings, noise 2) challenges in feature engineering 3) non-representative models.

1.1.2 Challenge 2: Application usability

The inference results are not 100% correct and therefore the design of the app should help overcome this inaccuracy. The app should still work if the readings become noisy or inaccurate.

1.1.3 Challenge 3: Power scarcity

Location sensing, other sensors, CPU computations, network activity and GPU usage all consume power which is a scarce resource and mobile and wearable devices. Applications should take into account the power scarcity.

1.1.4 Challenge 4: New Operational Mode

Instead of using a single-active application on device, in mobile sensing you have multiple concurrent background sensing applications that provide situation-aware services.

1.1.5 Challenge 5: Resource contention

Foreground activity is affected by resources used by the sensing application in the background, e.g. continuous vision sensing can affect the framerate of your application.

1.1.6 Challenge 6: Poor scalability

Things that work in research papers and in lab settings might not always scale to the real world with real users and real settings. Need to test on real users!

To summarize the challenges:

1. Inference accuracy
2. Application usability
3. Power scarcity
4. New Operational Mode
5. Resource contention
6. Poor scalability

1.2 Sensors

Possible applications of mobile sensing: 1) Mobile Activity Tracker that monitors exercise progress, reliable feedback on how much users move, gamifies it 2) Quantified self: Continuously track user activities and objectively summarize and visualize data, elicit positive behavioral changes

Sensors that we can use for these purposes

- Inertial sensors
  - Accelerometers: measure linear acceleration (m/s^2) in three different directions
  - Gyroscope: measure orientation and angular velocity
  - Compass: measure direction
- ...

1.3 Example application: Step detection

Initial idea: take 3-axis signal, convert it to a 1-axis magnitude signal (remove orientation), filter signal to remove noise and highlight the actual walking pattern, then detect steps by counting peaks

1.3.1 Extract signal magnitude

Take the magnitude of the accelartion vector as $\sqrt{x^2 + y^2 + z^2}$

1.3.2 Noise removal

Need to apply various pre-processing techniques to remove noise and make the data we are interested in easier to extract. Common techniques are:

- Moving average smoothing: sliding window, use average values of multiple adjacent samples.
- Exponential smoothing: gives more weight to more recent values, we assign exponentially decreasing weights to older values:

  $$s_1 = x_0, \tag{1}$$
  $$s_t = \alpha x_{t-1} + (1-\alpha) s_{t-1} \tag{2}$$

  for smoothing factor $0 < \alpha < 1$. Problems:
  - Averages out peaks in the data

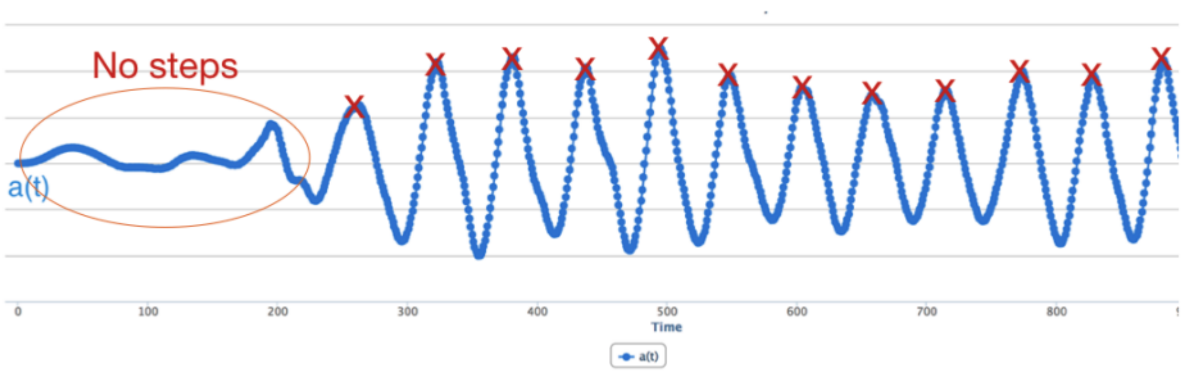


Figure 2: Filtered acceleration signal with highlighted peaks

  - Amplitude gets smaller
  - Peaks are lagging in time, slightly shifted to the right

- Median filtering: sliding window, takes median values of adjacent samples
- Frequency domain filtering: finding a good time domain filter can be hard so instead of time domain, focus on the frequency domain! Idea:

  Convert a signal to a weighted sum of sine waves, and remove all the waves whose periods are outside the range that you expect!

  Filters: low-pass filter, high-pass filter, bandpass filter (only lets through frequency band), notch filter (opposite of band)

1.3.3 Detecting steps

Count the highest peaks (see Figure 2). How do we accomplish this? We can take the derivative and look at when the signal changes from negative to positive.

2 Activity and Gesture Recognition

2.1 Activity recognition

Activity recognition: identifying the physical activity of a user from sensor data, e.g. walking, standing, sitting, jogging. We can use activity recognition to provide useful information and knowledge to the user.

Activity detection: single activity type
Activity classification: multiple activity types

It's easy to come up with simple heuristics but they are not always good: how do we determine good features/thresholds? What if user puts her phone in her bag, not in her front pocket (assumptions are broken)?

We can use machine learning techniques to learn to classify these activities from data.

2.1.1 Data Collection

First step is to collect sufficient (labelled) data for each activity, e.g. an hour of accelerometer data from a phone for each activity. Use this as the *training dataset*.

2.1.2 Feature extraction

Next we identify distinguishing features in the data. For instance for time domain features we can get aggregate statistics such as the average or the standard deviation. For frequency domain features we can look at periodic patterns or rhythmic behavior. More specifically:

Time domain features	Frequency domain features
Mean, median, variance, standard dev., min, max, range, zero-crossings, angle, angular velocity	dominant frequency, signal energy, etc.

2.1.3 Classifier training

Input features and ground truth data to a classifier which can learn to identify the most useful features in distinguishing between the different classes (activities). Example: decision tree which is a simple but very effective ML classifier.

Other useful ML techniques:

- Random Forest
- Support Vector Machine
- Naïve Bayes
- Hidden Markov Model
- Gaussian Mixture Model
- Neural Networks

2.2 Gesture recognition

Gestures are a natural way of interacting with objects and other people, they can be useful in certain contexts (i.e. while running, communicating with impaired people etc.).

What are the intuitive and accurately recognizable set of gestures? → Active research area in HCI.

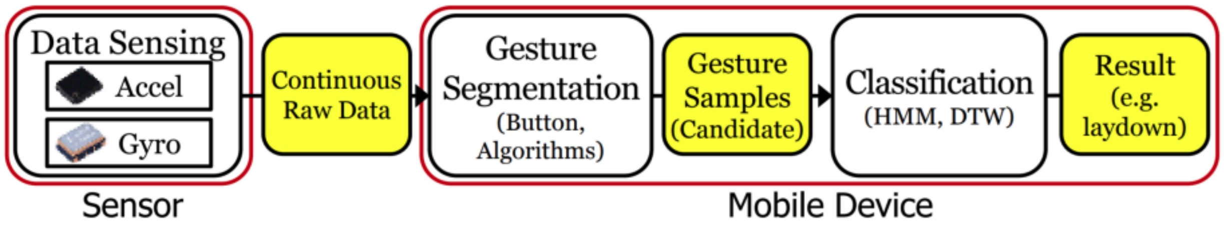

Data Sensing
Accel
Gyro
Sensor
Continuous Raw Data
Gesture Segmentation (Button, Algorithms)
Gesture Samples (Candidate)
Classification (HMM, DTW)
Result (e.g. laydown)
Mobile Device

Figure 3: Gesture recognition pipeline

Challenges are to provide **energy-efficient** gesture processing as well as to **accurately** detect and classify the hand gestures.

2.3 Paper: E-Gesture

E-Gesture: A Collaborative Architecture for Energy-efficient Gesture Recognition with Hand-worn Sensor and Mobile Devices

Contributions:

- Investigated the characteristics of sensors: accelerometer is mobility sensitive and energy-efficient while gyroscope is mobility-robust and energy-hungry
- Designed energy-efficient, mobility-robust gesture detection architecture. Idea:
  - triggering gyroscope by analyzing the accelerometer signal
  - adjusting accelerometer sensitivity by gyroscope validation
- Suggested two gesture classification architectures considering users' mobility patterns (HMM-based)

Challenge: moving creates mobility noise, makes it difficult to separate intended hand movements from mobility hand movements

Accelerometer is more sensitive to mobility while gyroscope is more robust to mobility.

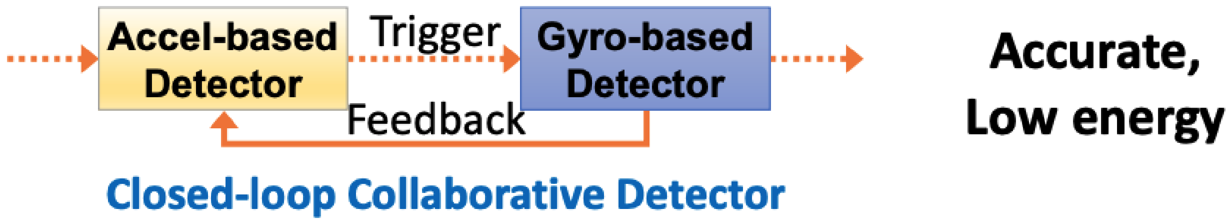


Figure 4: Closed-loop collaborative segmentation proposed in the E-Gesture paper

Idea: validate the gesture segments detected by the accel-based segmentor using the gyro-based segmentor. Furthermore, the accel-based segmentor is continuously adapted to dynamic changes in the mobility situations (feedback from gyroscope is used to update threshold for the accelerator).

Instead of a single HMM model for each gesture it is also possible to train a model for each gesture for a set of predefined mobility situations each (number of models = number of situations × number of gestures) or to train a model that adapts to different mobility situation through user feedback and online adaption (puts burden on the user).

2.4 Paper: Activity recognition from User-Annotated Acceleration Data

Problem statement: (physical) activity recognition from data generated by acceleration sensors. Most previous approaches did not work very well under real-world conditions. At the time of writing (2004) smartphones were not widespread so available data was limited.
Contributions:

- Activity recognition on user-annotated data
- Authors collect user-annotated data from 20 everyday activities in (semi-)naturalistic settings
- Data collection was performed while subjects were running an obstacle course
- Subjects were given freedom in how they perform the task and only note down start and end times
- Also collected data from participants doing the same activities in a more structured setting
- Train decision tree classifier to identify activities from features extracted using FFT (on sensor data)

⇒ training on more realistic data helps overcome accuracy gap between lab and real-world

2.5 Paper: Practical Human Sensing in the Light

Problem formulation: Sensing and tracking users indoors to reconstruct fine-grained user skeleton postures. Challenging due to furniture and different room layouts, user might move around the room freely. Prior work required user to wear a device or was based on cameras (possible privacy issues), was based on RF (susceptible to interference) or light-based but struggled with furniture and assumed a static user.

Contributions:

- A light-based sensing system that utilizes "regular" LED panels to reconstruct fine-grained user skeleton postures
- High accuracy, unobtrusive system that can reconstruct the skeleton at a high framerate (@ 40fps, higher than Kinect @ 30fps)
- Place lights on the ceiling, photodiodes in the floor. Idea: detect which light rays have been blocked and create a virtual shadow map on the ceiling, i.e. the shadow that would be seen if the photodiodes were emitting light (since we know which rays have been blocked)
- From these virtual shadow maps we can then identify the best-fit 3D skeleton posture
- Engineering: each LED has its own frequency, each photodiode measures light frequencies. We can thus infer which LED's light has been blocked (using FFT). Aggregate the virtual shadow maps created at each photodiode. Then reconstruct the user skeleton posture.
- Authors also propose an optimization algorithm to find the best LED positions for a room layout (with furniture)

2.6 Paper: BodyScan: Enabling Radio-based Sensing on Wearable Devices for Contactless Activity and Vital Sign Monitoring

Problem formulation: Wearable devices have multiple limitations: they require users to wear multiple IMUs (accelerometers, gyroscopes) to capture whole body movements as they can only capture data on the part they are attached to, microphones and image sensors capture privacy-sensitive data and physiological devices (respiration, EEG) require tight skin contact. To overcome these limitations we need a **contactless** and **privacy-preserving** approach.
Contributions:

- The authors introduce radio as a new sensing modality for human activity recognition and to detect vital signs
- First wearable sensing system that uses only a single pair of on-body radio transmitter and receiver to monitor a wide range of human activities and vital sign
- This approach is both contactless and privacy-preserving
- Assumes the user wears a wristband (e.g. of a smartwatch)
- This special ESP wristband can act as a radio transmitter and it has two conical horn directional antennas mounted on the top and the bottom to capture activity from both the upper and lower body parts of the wearer
- For classification they use channel state information, i.e. measurements about how the radio waves propagate combined with denoising techniques (low-pass, PCA)
- Dominant (low) frequency in the chest movements is the breathing rate → do FFT and obtain frequency with highest peak which is the estimated breathing rate

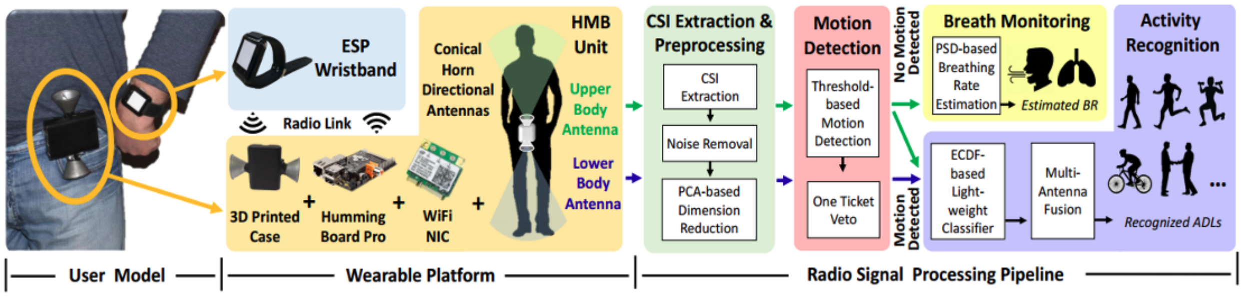


Figure 5: System architecture

3 Location Sensing

3.1 Importance of location

Location is the primary context of a mobile user. It encodes a rich amount of information. Many applications use location: Foursquare, a location-based social network, sports trackers taht allow sharing routes and stats, Pokémon Go, Kidgy which keeps track of your kids via geofencing, Waze which crowd-sources traffic information from other drivers using the app in order to give up-to-date traffic information and good rereouting, traffic flows can be important to better plan and manage cities (urban planning). Indoor location can be interesting as well: indoor navigation, space planning, queue detection, toilet usage monitoring.

3.1.1 Localization on smartphones

Different sensors and their accuracies:

Sensor	Accuracy
GPS	10m
Cellular	100m
WiFi+GPS+Cellular	10m - 100m

3.2 Global positioning system (GPS)

GPS satellites are space vehicles (SVs), they are solar powered and take corrections from (ground) control stations. They output X, Y, Z and t data streams.

There are 24 satellites in the system plus three backup satellites (for a total of 27 satellites), rotating around the earth at an elevation of 12 000 miles, each making two orbits around the earth a day. At any given time there are **multiple visible satellites** from any point of the earth.

In theory we can use 3 satellites to acquire a 3D location (3 equations):

$$A_{\text{Geo}} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \end{bmatrix} \tag{3}$$

However, this gives us 300km of error due to unsychronized clocks. Due to this we add a new unknown parameter δ and use a 4th satellite to estimate both the location and this parameter:

$$A_{\text{Geo}} \begin{bmatrix} X \\ Y \\ Z \\ \delta_{clk} \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \end{bmatrix} \tag{4}$$

Thus, four satellites are needed for an accurate 3D position. In reality, due to additional sources of error (satellite positions, weather, multipath [signal bounces off somewhere else but still reaches destination, increases length of time], timing errors) more satellites are preferable.

3.3 Cell tower-based localization

3.3.1 Cell ID

Each base tower has a certain range so any device connected to it has to be in this tower's base station coverage.

3.3.2 Angle of arrival (AOA)

Measures the angle from which a signal arrives, reduces the possible area the device can be located in (think to a slice in a cake).

3.3.3 Time of Arrival (TOA)

Measure the time of arrival of the signal, allows us to infer the distance of the device and therefore the device has to be located on a ring of that distance.

3.3.4 Hybrid (TOA+AOA)

Hybrid allows us to infer the position of the device as a "ring on a slice", a much smaller area than the entire area covered by the base tower and thus much more accurate.

3.4 Basics of indoor localization

Can be distinguished by their underlying technologies: IR, RF, ultrasonic, Wi-Fi, BLE, UWB, RFID

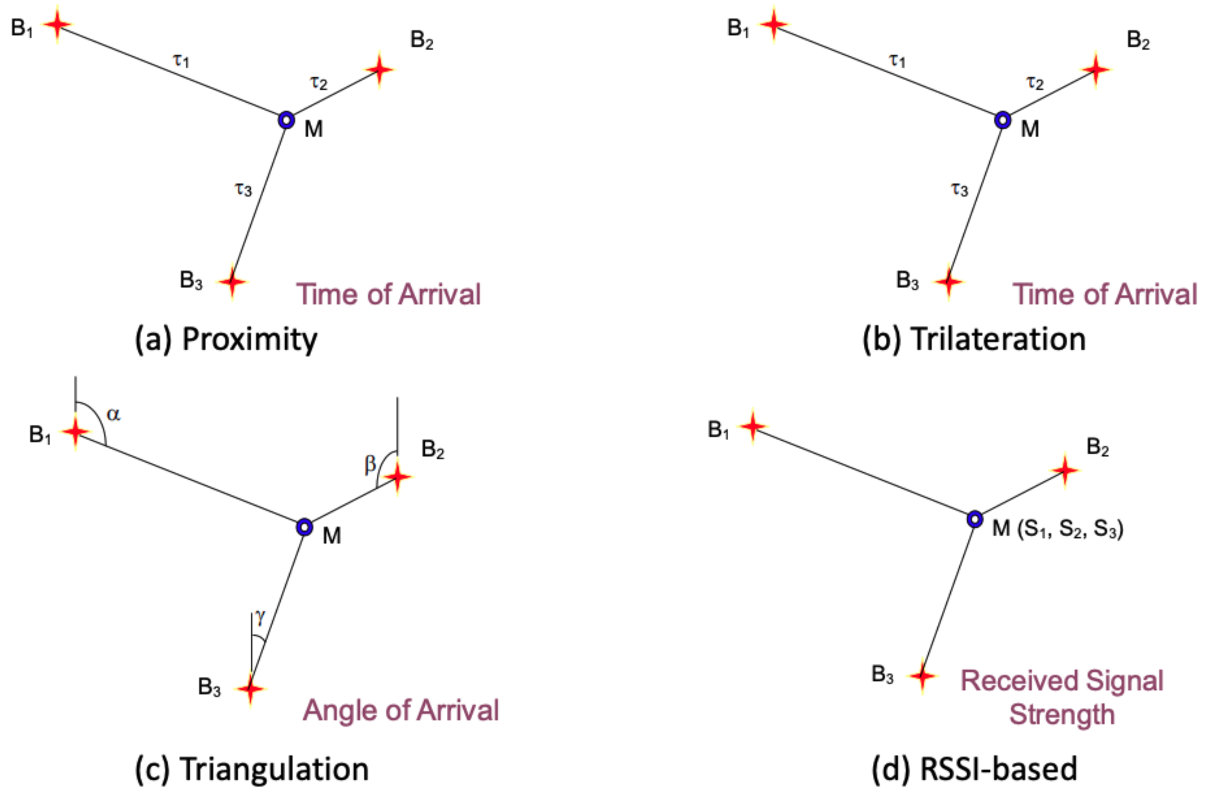


Figure 6: Common Localization Approaches

See Figure 6. Common indoor localization approaches are based on

- proximity: measure distance between device and reference points (similarly to GPS). Then use the location of the closest reference point as its location. Useful when you want to know which room the device is in.
- trilateration: measure the distance between device and reference points (as in GPS), again 3 reference points needed for 2D location and 4 reference points for 3D location.
- triangulation: measure which direction the reference signal comes from (i.e. the angle of arrival) to estimate its location
- received signal strength-based (RSSI-based): send out signal of known strength, use received signal strength and path loss coefficient to estimate distance
- RSSI fingerprint-based localization: compute RSSI fingerprints, i.e. signal strength characteristics at selected locations, then use prebuilt fingerprints to determine the location

3.5 Paper: The Cricket Indoor Localization System

Idea: ultrasonic and RF signals travel at different speeds (that we know). Send both a ultrasound pulse and RF data (i.e. the space name) from a beacon to the listener. The listener can estimate the distance from the beacon by measuring the time gap between the receipt of RF and ultrasonic signals.

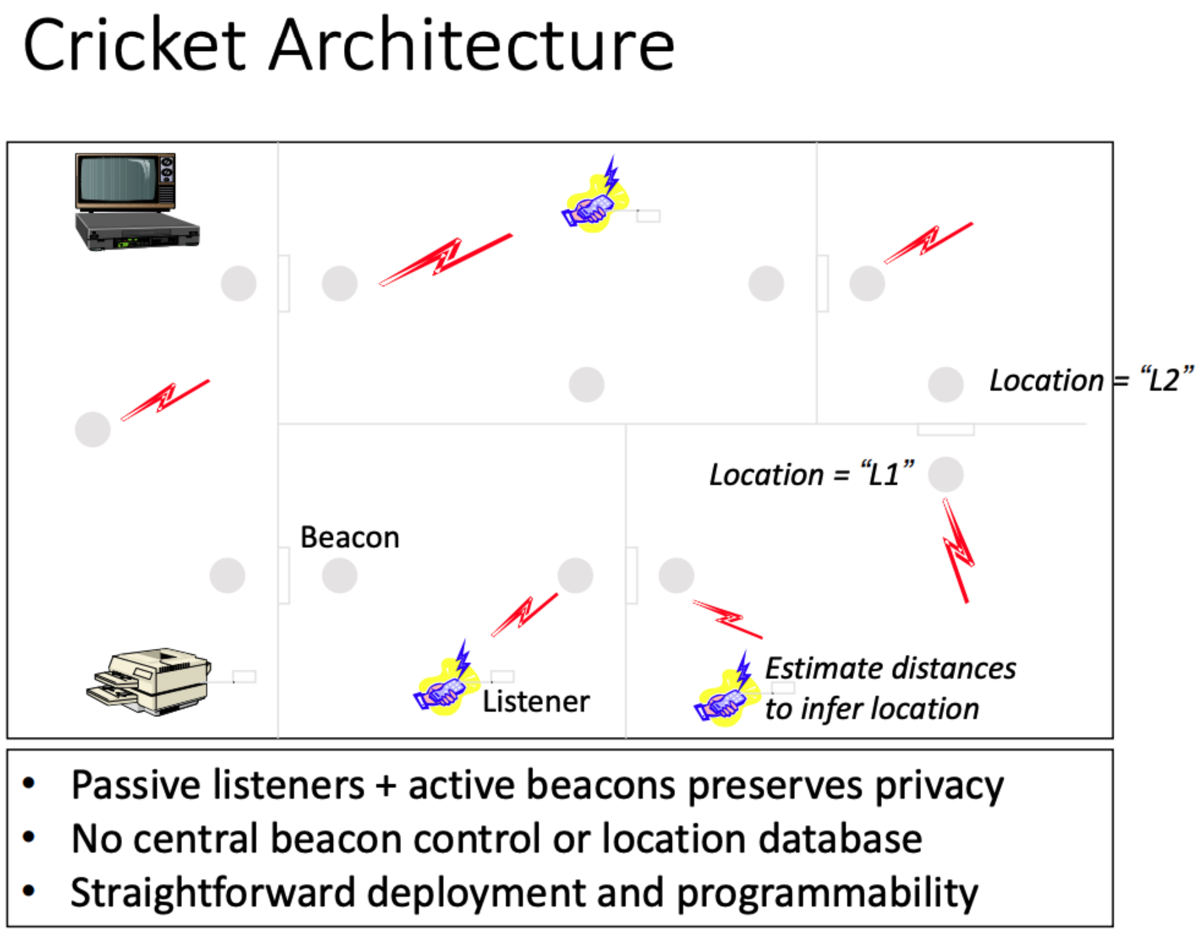
- A time gap of x ms roughly corresponds to a distance of y feet from beacon
- Velocity of ultra sound $\ll$ velocity of RF

Uncoordinated beacon transmissions can cause interference problems as you don't know which ultrasonic pulse belongs to which RF signal. We can reduce changes of concurrent beaconing through *carrier-sense* and *randomized transmission*.

Furthermore to reduce stray signal interference, only transmit ultrasonic signal while RF signal is still being transmitted (but start later with time gap). Also RF range should be larger so that if listener can hear ultrasound they can also hear RF (so no more interfering ultrasound). Also important: beacon deployment so closest beacon actually represents location well.

3.6 BuSCOPE : Fusing Individual & Aggregated Mobility Behavior for "Live" Smart City Services

Problem formulation: Given historical and real-time commuter data, what kind of insights can we gain? Can we predict where a passenger will disembark in real-time?



Cricket Architecture

- Passive listeners + active beacons preserves privacy
- No central beacon control or location database
- Straightforward deployment and programmability

Figure 7: Cricket architecture

- The authors introduce a system that can predict where a user is going to disembark by measures of support (i.e. fractions of trips starting at a given boarding stop) and a measure of confidence (probability of disembarkation at a stop in question given the boarding stop, i.e. the fraction of trips originating from the boarding stop and ending at the stop in question)

- The authors analyze general traffic patterns and find that there are sink nodes where a lot of travelers disembark and that there is a non-negligible amount of irregular travelers

- They introduce a system which can analyze the traffic and predict where travelers are getting off in real-time

- Authors propose two applications for this system:

  1) Last mile demand generator which improves the last-mile commuting experience by providing robotaxis where users are predicted to disembark: predict highest

confidence bus stop for regular travelers and highest aggregate flow-based probability from starting point for irregular travelers

2) Neighborhood event predictor which improves the utilization of city resources by providing buses, taxis where they will be needed: compute aggregate anomaly scores, i.e. count passengers that deviate from the prediction

3.7 Paper: No Need to War-Drive: Unsupervised Indoor Localization

Dead-reckoning: computing the motion trajectory of a mobile phone. Problem: accumulation of noise leads to diversion from the (ground) truth.

Problem formulation: Indoor localization requires high location accuracy. Pervasive Wi-Fi systems are accurate but at a high cost (mostly in the form of signal calibration).

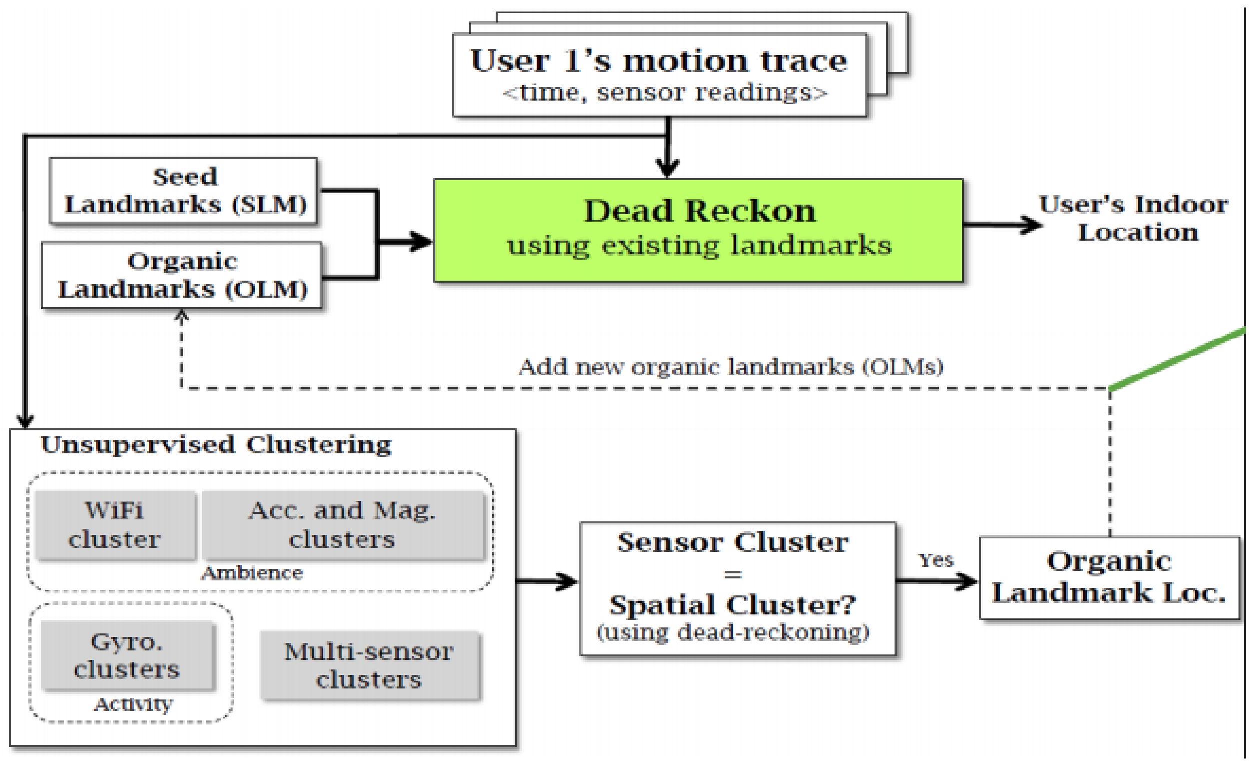


Figure 8: Architecture of the UnLoc system

- The authors propose an unsupervised indoor localization scheme that bypasses the need for war-driving
- Certain locations in an indoor environment present iden- tifiable signatures on one or more sensing dimensions (i.e. elevator imposes a distinct pattern on a smart- phone's accelerometer; a corridor-corner may overhear a unique set of WiFi access points)

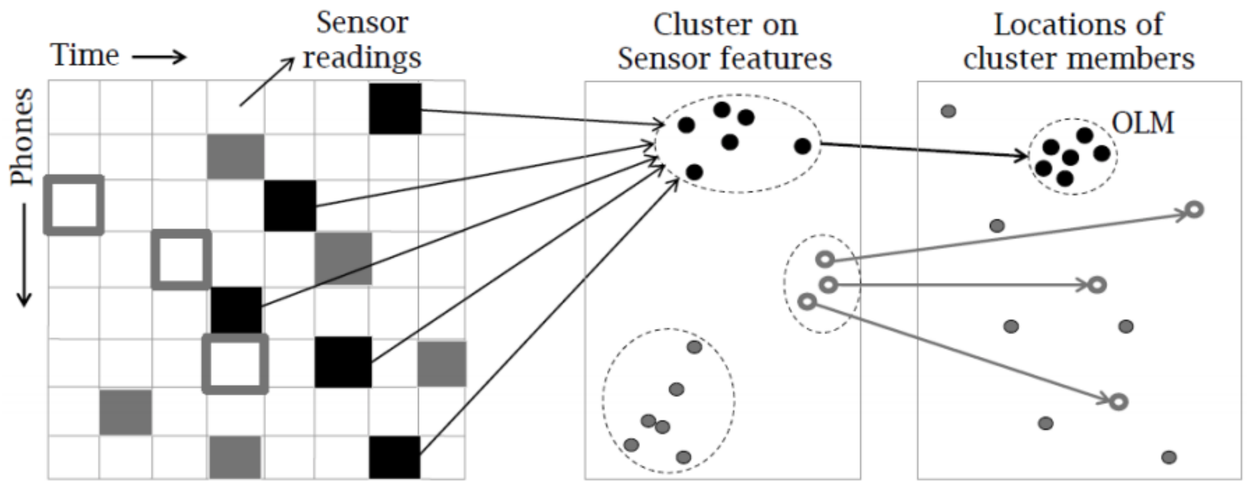


Figure 9: Finding new organic landmarks

- The authors suggest exploiting these *landmarks* to localize the device: use dead-reckoning to track the device location and recalibrate with landmarks when possible
- Additionally to these seed landmarks, the authors propose a scheme to naturally extend these with organic landmarks: cluster the sensor features and if these clusters also cluster in the spatial domain then add them as an organic landmark

Weaknesses:

- Limited to training device since the signals from the same location will vary across devices, hasn't been tested on multiple devices
- Using many sensors will consume a large amount of energy

3.8 Paper: RADAR: An In-Building RF-based User Location and Tracking System

This paper takes a WiFi fingerprint-based approach. Goal: track the location of mobile nodes inside a building using standard WiFi infrastructure.
Method does not need any additional hardware:

- Training phase: Collect $< \text{Location}, \text{Signal} >$ ground truth
- Model: Build some model of relationship between location and WiFi signal strength
- Test phase: Observe the signal readings and use the model (reverse it: signal strenth *rightarrow* location) to obtain the location

Disadvantages:

- Requires creation of a database
- Changes in environment require re-fingerprinting

With n different WiFi signals that are being received, each fingerprint at a location l has average signal strengths $s_1, \dots, s_n$ and can be considered a point in this n-dimensional Euclidian space. To get a location we can now compute the signal strengths to all WiFi APs and find the fingerprint that is closest (Euclidian distance) and get its location. Optional: compute the centroid of the k nearest landmarks.

3.8.1 Extension 1: Model-based approach

Utilize a model to predict the signal strength at different points without actual empirical measurements:

$$\text{RSSI} = f(\text{distance})$$

Advantage: reduced training effort, but multipath errors and loss due to obstructions can be a problem

3.8.2 Extension 2: Hyperbolic Radar

Aims to tackle limiting assumptions in RADAR, i.e. device diversity (different radios, antennas, measurement errors).

Idea: instead of raw signal strength, simply use the normalized (ratios) of signal strengths. These ratios are more stable across different clients.

Disadvantage: increased size of fingerpint table, longer computation times.

3.8.3 Extension 3: Continuous Tracking with Radar

Exploit the temporal history of movement through trajectory tracking. Use a Viterbi-like algorithm for *maximum likelihood estimation*:

- Maintain a depth of h consecutive samples
- Edge weights represent distances between location samples
- Select path that minimizes the sum of edge weights (since random jumps are not feasible)

Disadvantage is the lag of h time slots before location is estimated.

3.9 Paper: Horus: A WLAN-Based Indoor Location Determination System

Basic method:

- Offline phase: radio map: collect signal strength histograms
- Online phase: Do Bayesian-based inference

3.10 Paper: Geo-Indistinguishability: Differential Privacy for Location-Based Systems

Problem formulation: Location-based services provide a service to a user based on their location. e.g. recommending nearby restaurants. However, location information is often extremely sensitive and may reveal private information (home and work address, sexual preferences, political views, health problems etc.).
Contributions:

- The authors introduce a notion of spatial indistinguishability based on differential privacy to give mathematical privacy guarantees to the user
- ϵ-geo-indistinguishability: user specifies requirements (radius r and privacy level l), then:

  A mechanism K satisfies ϵ-geo-indistinguishability if and only if

  $$d_P(K(x), K(x')) \leq \epsilon d(x, x') \forall x, x'$$

  i.e. two locations in a given radius r will produce similar outcomes (different by at most l)
- Also propose a mechanism to achieve this notion: Planar Laplace mechanism. The probability of generating x should decrease exponentially with the distance from the actual location x_0. This mechanism maps the location domain as a discrete Cartesian plane. They add noise to this plane with a novel way to sample from planar Laplace distributions.
- The authors describe several LBS applications that can be enhanced by their method

3.11 Paper: Trajectories of Depression: Unobtrusive Monitoring of Depressive States by means of Smartphone Mobility Traces Analysis

Problem formulation: Depression is a serious problem. How can we utilize mobile devices to monitor depression, specifically how can we use location information to accomplish this?
Contributions:

- The authors develop a smartphone application that can periodically collect the location of users and their answers to daily questionnaires which can quantify their depressive mood (on that given day)
- Using this system they show that there is a significant correlation between mobility trace characteristics and the depressive moods
- Mood traces:
    1. Collecting user locations: mobility trace is a sequence of stops and moves

       $$\text{Place} = \langle \text{ID}, t_{\text{arrival}}, t_{\text{departure}}, C_{\text{lat+long}} \rangle \tag{5}$$
       $$\text{MT}(t_1, t_2) = (\text{Place}_1, \text{Place}_2, \ldots, \text{Place}_{N(t_1,t_2)}) \tag{6}$$

       Mobility metrics:
        * the total distance covered
        * the maximum distance between two locations
        * the radius of gyration (coverage area)
        * the standard deviation of displacements, i.e. distance between one place and the subsequent one
        * the maximum distance from home
        * the number of different places visited
        * the number of different significant places visited (significant: top 10 most visited places)
        * routine index: the average difference between the mobility behavior of the user in $[t_1, t_2]$ and the mobility behavior of the same user in other days in the same (daily) time interval
    2. Collecting answers to PHQ depression test
    3. Training machine learning model with the collected data
    4. Monitoring depression without user interaction based on their movements
- The proposed app is energy-efficient: only sample location occasionally, decide through use of state machine (static/moving/undecided) with activity data
- Based on these findings they present models (personalized/general) which can successfully predict changes in the depressive mood of individuals by analyzing their movements

3.12 Other useful methods

- Unloc: sensors + dead reckoning
- Continuous tracking: particle filters
- Continuous tracking: enhanced Viterbi
- Visible light-based localization

4 Pervasive Healthcare & Physiological Sensing

Opportunities:

- Doctors aren't available everywhere and pervasive healthcare might help alleviate problems due to this
- Daily well-being tracking and preventive healthcare (think Apple Watch, Fitbit)
- Non-life threatening but life-impacting issues, chronic diseases: diabetes, arrhythmia, fatty heart, autism, ADHD

Wearable and IoT devicdes may measure various physiological signals that could be used for this such as cardiac rhythms, breathing, sweat, brain waves, gestures, muscular contractions, eye movements.

4.1 Heart monitoring using PPG and ECG

PPG. A non-invasive technique for measuring blood volume changes in the blood vessels close to the skin. Popular method for extracting the heart rate and oxygen saturation. Can even be measured with built-in cellphone camera without any additional hardware.

How it works. LED emits green light and a photodiode captures the reflected light levels. Detects the heart rate by measuring the differences in light absorption from the skin, bones and blood.
Heart rate calculation method:

1. Get the average green intensity per frame
2. Use a peak detection algorithm to find all the cardiac peaks in the signal (highest average green intensity in a fixed window ≈ 0.7 seconds
3. The time difference between consecutive peaks is cimputed. This time difference is known as *inter-beat interval* (IBI) or *R-R interval* (RRI).

4. Heart rate HR is estimated from the RRI: HR $= \frac{60}{\text{RRI}}$

Breathing rate calculation method: Heart rate increases during inspiration and decreases during expiration.

1. Get the average green intensity per frame
2. Perform FFT
3. The second dominant frequency will match up to the respiration rate

Problems. Accuracy of these heart rate readings may not always be high and this can negatively impact the healthcare application's implications on the user's health status. The accuracy can for instance be impacted by vibration caused by motion artifacts causing irregular light intensity readings at the photodiode.

Idea: Use accelerometer or light intensity to predict whether the sensor is currently predicting the heart rate accurately so the application can selectively use the measurement according to the accuracy.

ECG. An Electrocardiogram or short ECG is a recording of the electrical activity of the heart. With each heartbeat an electrical signal spreads from the top of the heart to the bottom. As it travels the signal causes the heart to contract and pump blood. This process repeats with each heartbeat. It is measured by using two or more electrodes at different points on the chest (or on two opposite body parts from the chest) which measure electrical activity between them.

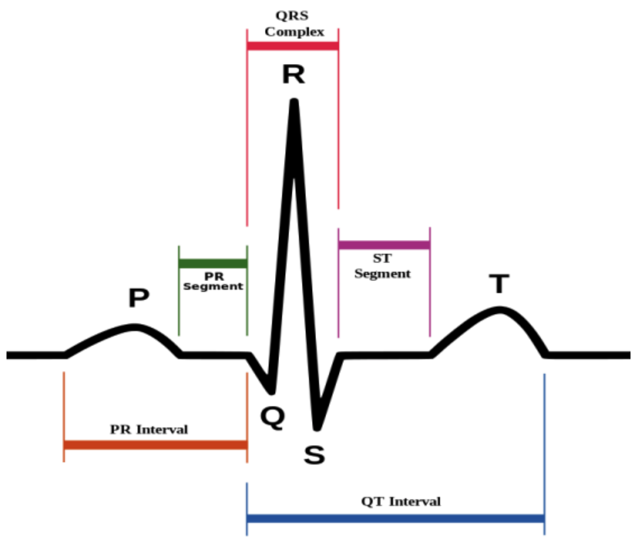


Figure 10: The different peaks in an ECG signal.

How it works. Obtaining heart rate from ECG is straightforward: like in the step detector from section 1, you simply look for a change in the slope from positive to negative to identify the peaks. Then look at the sequence of peaks. Each RR interval (RRI) corresponds to the time between two successive heartbeats and as previously: $\mathrm{HR} = \frac{60}{\mathrm{RRI}}$. The signal includes a number of different peaks (P, Q, R, S, T). Other intervals such as the PR interval, QRS interval, QT interval, ST interval may be intersting as well to detect abnormalities in the heart. Can also be used to detect stress.
Limitations of wearables:

- Reliability of sensing
- Motion artifect
- Washing durability
- Sensor requires skin contact
- Burden to wear and measure continuously

Heart rate variability (HRV) extraction.

1. Filter out non-contact signals
2. Detect noisy section
3. Drop time windows that are hard to correct
4. Detect RR intervals
5. Correct QRS peaks
6. Interpolate missed QRS peaks

4.2 Other physiological sensing techniques

Electrodermal Activity (EDA). Electrodermal activity refers to electrical changes at the surface of the skin. When people experience emotional arousal, increased cognitive workload or physical exertion, the brain sends signals to the skin to increase the level of sweating. The person may not feel any sweat, but the electrical conductance increases in a measurably significant way. EDA can be used to examine implicit emotional responses that may occur without conscious awareness or are beyond cognitive intent.
Main components of EDA:

- Skin conductance level (SCL):
  - Slowly changes over the course of minutes

  - Reflects general changes in autonomic arousal

- Skin conductance response (SCR):
  - Phasic component
  - Faster changing elements of the signal (changes within seconds)
  - Corresponds to sudden events (e.g. when startled)
  - Several useful features can be extracted for classification:
    - Latency: amount of time between stimulus and rise of the wave
    - Rise time: how long it takes for the skin conductance to shoot up to its peak
    - Amplitude: height of the SCR
    - Recovery time: amount of time it takes for the wave to fall back to a certain level of its amplitude

Electroencophalography (EEG). EEG signals are recordings of brain signals that are traditionally captured with a huge array of electrodes on the scalp. Recent EEG headbands have fewer electrodes to capture a subset of the EEG signal. Allows for classification of various states (aroused, relaxed, asleep, etc.).

Electromyography (EMG). Captures the electrical signal produced by skeletal muscles. It is useful to detect various gestures from fingers (using an armband), and to enable gesture-based interaction with computers.

4.3 Paper: VitaMon: Measuring Heart Rate Variability Using Smartphone Front Camera

Problem formulation: Heart rate variability (HRV) measures the fluctuations in the interval between consecutive heartbeats. It is an important measure that can be used to diagnose various diseases and to measure stress. Previous HRV measurement techniques however require additional instruments such as electronic or optical sensing devices that are not readily available.
Contributions:

- The authors propose a HRV sensing method that uses only the smartphones front camera, something readily available to many people.
- Can be done while phone is used for other tasks, e.g. to measure stress during a real-time game
- They manage to extract HRV from low-framerate (15fps) front camera video:

- Take multiple PPG readings from different facial areas (opposed to just one from previous approaches)
- Exploit that the signal in different areas are phase shifted, i.e. it takes some time for the blood to reach the forehead (vs. the chin)
- Build a CNN-based deep learning system that can predict exact heartbeat timing from facial video by exploiting this phase shift
- Multiple readings also help alleviate problems from noise in the video

4.4 Paper: StudentLife: Assessing Mental Health, Academic Performance and Behavioral Trends of College Students using Smartphones

Problem formulation: Explore some questions about student life and stress. Why do some students do better than others under similar conditions? What is the impact of stress, mood, workload, sociability, sleep and mental well-being on education performance?

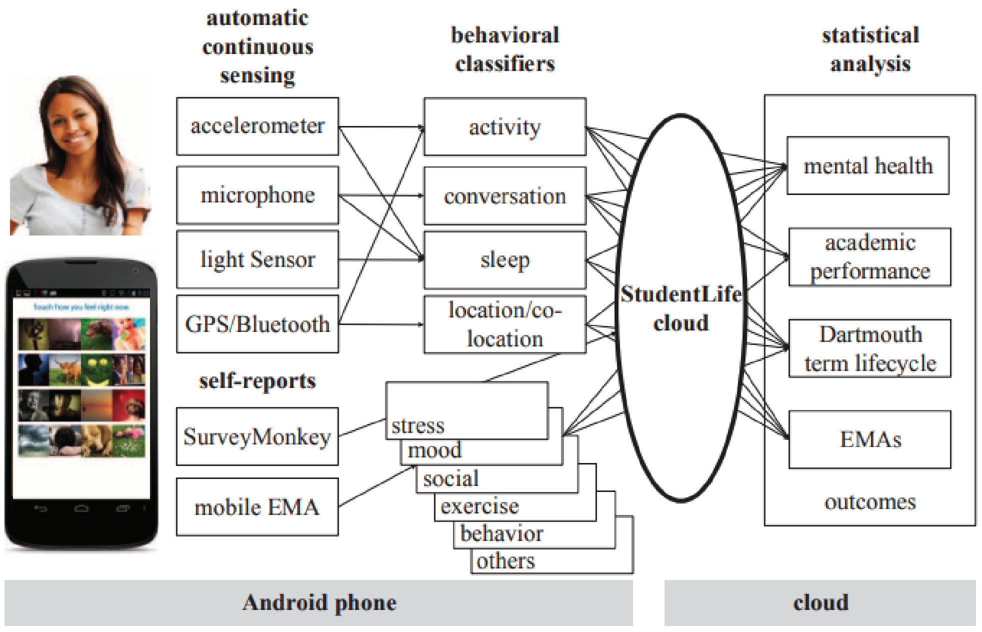


Figure 11: StudentLife app, sensing and analytics system architecture

Contributions:

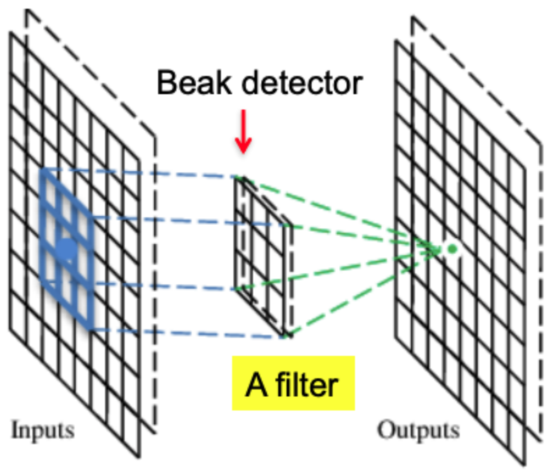
- The authors carry out a study using smartphones carried by students to collect a dataset to answer some of these questions
- They develop a StudentLife app and sensing system:

- Performs automatic and continuous sensing:
  - Activity detection: daily activity duration (threshold-based) by adding up all 10 minute active periods, outdoor mobility (distance traveled) using GPS samples, indoor mobility using WiFi scan logs
  - Conversation detection: captures number of conversations and their duration, two-state HMM to infer whether the user is around conversation
  - Sleep detection: sleep duration as well as bedtime and wake up time, sleep classifier from light features, phone usage features (i.e. phone locked), activity (e.g. stationary) and sound features from the microphone
- Performs surveys to capture the students mood and feelings: user selects picture that represents their feelings (photographic affect meter or PAM), short survery (pop-up EMA)

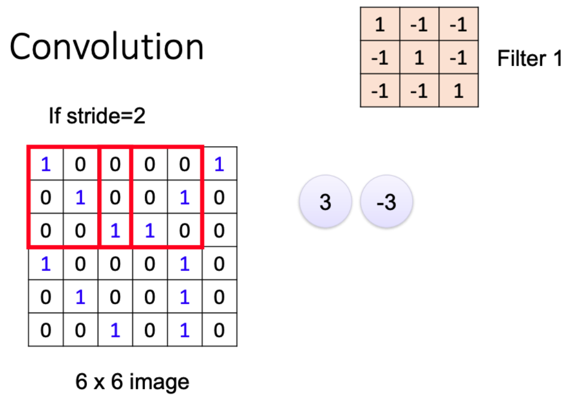
5 Mobile and Embedded Machine Learning

Deep Learning: extract features automatically, learn end-to-end (i.e input data to output). The more data the higher the accuracy. Can be applied in many fields such as self-driving cars, face recognition, speech recognition, games (chess, go) and mobile sensing.

5.1 Introduction to Convolutional Neural Networks



(a) An example of a CNN filter



(b) An example of a CNN convolution computation

Figure 12: The building blocks of convolutional neural networks

A Convolutional Neural Network (CNN) is a neural network with *convolutional layers* which have a number of filters that do convolutional operations.

Filters allow us to learn features from the image. Each convolutional filter corresponds to a channel. By then using max pooling (i.e. subsampling the image) we can reduce the number of parameters necessary to characterize the image. These steps can be repeated many times (and are in modern CNN architectures such as ResNet).

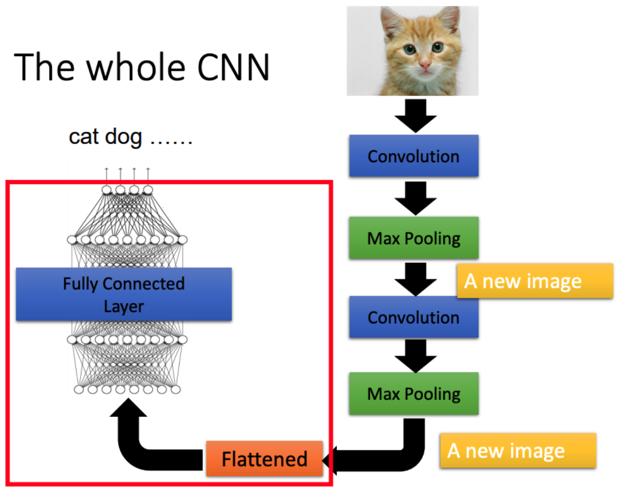


Figure 13: An example CNN architecture

5.2 Paper: DeepMon: Mobile GPU-based Deep Learning Framework for Continuous Vision Applications

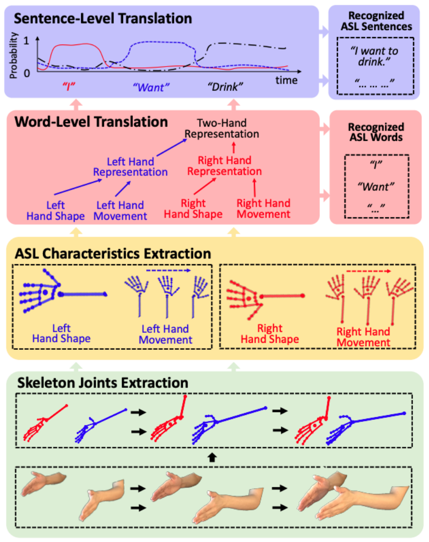
Problem formulation: Can we support fully-disconnected DNN-based inference purely on the mobile device (i.e. without offloading to the cloud)?
Contributions:

- DeepMon is a framework that aims to deal with these challenges in mobile devices
- Supports low-latency execution of CNNs on commodity mobile devices using mobile GPUs
- Supports multiple GPU architectures & mobile OSes
- Supports existing trained models from multiple freameworks (Caffe, Matconvnet, Yolo, Darknet)
- Overcomes several challenges in mobile devices:
  - Mobile GPUs are weak: the execution latency is too high to support continuous vision

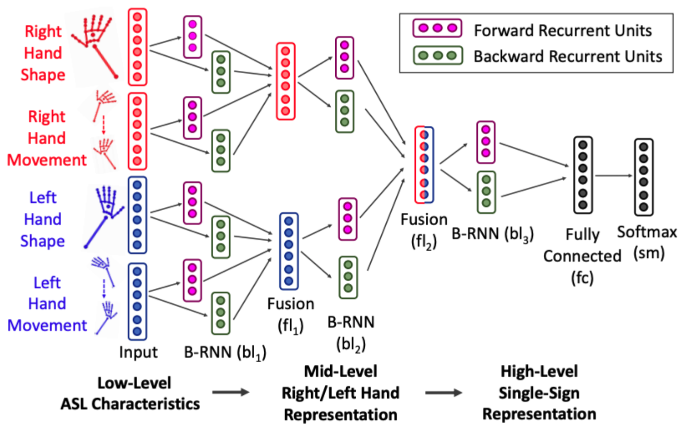
    - Architecture of mobile GPUs is different: GPU-based optimizations won't work

- The authors identify the latency bottleneck to be convolutional layers which consume around 90% of the time
- Reduce memory usage by doing convolution operations directly on the input (avoids matrix-building overhead) and various mGPU-aware optimizations
- Convolutional caching to resue results from previous convolution operations for regions that remained static (i.e. background regions)
- Decompose the large convolution layer into a sequence of smaller ones to reduce computation costs (Tucker-2 decomposition)

Problem formulation: There is a communication barrier between deaf people and people with normal hearing ability. Can we utilize mobile devices to do translation of sign language? Existing solutions do not really attempt to translate entire sentences at once but rather focus on translating individual signs one-by-one. They are also intrusive or contrained by resolution or ambient light conditions.



(a) System architecture



(b) Architecture of the hierarchical bidirectional deep recurrent neural network

Figure 14: Overview over the components of DeepASL

- The authors introduce a system that can capture ASL signs using infrared light in a non-intrusive manner, then process the captured skeletal information using domain knowledge and finally use a hierarchical bidirectional RNN (HB-RNN) as well as a probabilistic framework (Connectionist Temporal Classification or short CTC) to make the final sentence-level translation

- They overcome various challenges in transforming the raw skeleton joint data collected by the *Leap Motion* infrared sensing system:
  - **ASL characteristics extraction** from raw skeleton joint data: raw skeleton joint data is smoothed over the temporal sequence using *Savitzky-Golay filter* (sliding window weighted sum), extract *hand shape information* i.e. the relative distance of joints to one another zero-centered on the palm center of the right hand as well as *hand movement information* which is the spatial displacement of a joint between two consecutive frames (i.e. how far it has moved)
  - **ASL characteristics organization** into a high-level representation by means of a RNN model: feed characteristics into
  - **Use of bidirectional RNN models** to overcome similarity of signs and to incorporate the future trajectory information at the sign-level
  - **ASL sentence-level translation without segmentation** of the sentence but rather translating it as a whole using CTC

5.3 Paper: DeepEar: Robust Smartphone Audio Sensing in Unconstrained Acoustic Environments using Deep Learning

Problem formulation: What kind of information can we infer from the microphone? Can we use audio sensing to accurately identify activities in diverse acoustic environments? Previous audio sensing models (shallow) are not robust to different acoustic environments.
Contributions:

- The authors propose a mobile audio sensing framework built with DNNs that can do accurate audio sensing for various inference tasks with diverse acoustic environments
- They introduce an unsupervised pre-training scheme utilizing unlabeled data from a large amount of locations. This helps the model generalize better across a diverse set of environments.
- After pre-training they do supervised fine-tuning of the model with the data relevant for the respective task
- This model is feasible for use on mobile device as it directly utilizes the DSP (digital signal processor) and is able to perform continuous sensing with acceptable levels of energy and latency

6 Power-Aware Design of Mobile Sensing Systems

Mobile sensing can draw a lot of power if the sensors are utilized carelessly. When designing such a system, we should of course keep in mind the resource constraints of mobile hardware.

6.1 Power Optimization Strategies

- Duty Cycling
  - Significant amount of power is consumed if a sensor is not powered off
  - Control the sensing period to fit the energy budget (turn sensor on and off as needed)
  - Sensing period T makes an accuracy-energy trade-off, need to choose T carefully but this can be difficult
  - T can either be chosen statically or dynamically in an adaptive manner
  - Need to consider: some sensors need time to turn on and off
- Hierarchical Sensing
  - Use energy-efficient sensing modules to trigger power-hungry sensing modules
  - Use simple pre-processing logic to filter out uninteresting sensor data and apply computation-intensive logic only on the potentially meaningful data
  - Example: Rate adaptive GPS, uses accelerometer to detect user motion and only triggers GPS when there is movement
  - Example: E-Gesture, use accelerometers as cheap first-order filter, turn on gyro only when accelerometer indicates potential gestures
- Piggybacking
  - (Additional) Power consumption of sensing can be significantly lower while other apps are running, "piggyback" on their sensor usage
  - Useful technique to reduce overall power draw
  - Example: Piggyback CrowdSensing, opportunistic crowdsensing system that reports useful sensor data without affecting users too much
- Adaption
  - The same user context can be monitored in multiple ways, all of which consume different levels of power (e.g. different sensor combinations, different processing logics)
  - We can alternatively use such different methods given the power budget
  - Again we have an accuracy vs. energy trade-off
  - Example: Orchestrator, enables multiple applications to effectively use system resources and multiple devices dynamically
- Cooperative Sensing

    - Cooperation between devices on a more local level, e.g. Bluetooth, can save power
    - Devices can exchange information about location, events, etc. when they connect
    - There are enough chances in daily live for cooperative sensing as a lot of time is spent around acquaintances or strangers
    - Number of contexts that can be shared
        * Spatial context: location, place, ambient sound, crowdedness, noise-level, mood, pollution, temperature, ...
        * Social context: companion, relationship, interaction type (discussion, meeting, conversation, lecture), ...
        * Personal context: activity (walking, standing, etc.), gesture, health (heartbeat, gait, etc.), emotion, ...
    - Example: CoMon, a cooperative ambience monitoring platform

- Prediction
    - Predict when sampling will be needed or when current prediction will become inaccurate (e.g. during dead-reckoning)
    - Example: EnLoc, prediction of regular movement patterns. Predict user location when movement is along habitual paths to avoid unnecessary sampling. When prediction is unreliable, sample as normal.

- Voltage Scaling

6.2 Paper: Sparsification and Separation of Deep Learning Layers for Constrained Resource Inference on Wearables

Problem formulation: Deep learning models require a significant amount of power and computing resources which can be a limiting factor when intending to use these models on mobile devices (even more so on wearable and embedded devices). Due to these constraints the majority of mobile applications using deep learning models remain cloud-assisted which both introduces latency (rendering it useless for real-time applications) and exposes user information to these services.
Contributions:

- The authors propose a new sparsification approach for fully connected layers and convolutional kernels thereby lowering the resource requirements and making it possible for larger DNNs and CNNs to run on mobile and embedded hardware

- In particular, they leverage theory from sparse dictionary learning and make it possible to reduce the size of weight matrices, a major limiting factor in deep learning factors, essentially splitting these linear layers up into smaller layers. By enforcing sparsity constraints on the factorization process they manage to reduce the number of computation while only sacrificing a small amount of accuracy.

- Due to the smaller matrices being used less computational and memory resources are required. They also provide theoretical bounds on errors from the original models.
- In addition, they also introduce a novel kernel separation technique to improve inference time and power consumption of CNN models. This technique essentially replaces a larger convolutional layer with filter $\mathcal{K}$ with two successive convolution layer (of smaller size) and two filters $\mathcal{V}$ and $\mathcal{H}$.
- These techniques make it possible to run large-scale deep learning models on mobile hardware for the first time
- Based on this approach, the authors implement a prototype consisting of a compiler, a runtime framework and a convolution separation runtime and carry out experiments on different tasks to show the efficacy of their approach

6.3 Paper: Energy Characterization and Optimization of Image Sensing Toward Continuous Mobile Vision

Problem formulation: Image sensors are power-hungry. As a result, CMOS image sensors are often uses nowadays, since they are comparatively low-energy and low-cost. Goal: identify comprehensive energy characteristics of CMOS image sensors: relation between power and image quality, systematic methods to reduce power consumption, hardware improvements that could reduce power consumption.
Contributions:

- The authors introduce three mechanisms to improve the energy efficiency for continuous mobile vision:
  - Optimal clock scaling: the optimal clock frequency of the image sensors can reduce power consumption
  - Aggressive standby mode: similar to repeating single frame captures
  - Architectural modifications

7 Mobile Cloud and Edge Systems

Rich apps are often hindered by resource-constrained mobile devices (battery, CPU, memory, ...). How can we seamlessly offload compute-intensive parts to the cloud to do complex computations or save power? One recent trend in the field of cloud computing is the so called *edge computing* where cloud servers which are "far away" are replaced with *edge* *servers* which are located in the network edge and are thus much closer to the device thus allowing for lower latency computations.

7.0.1 Paper: MAUI: Making Smartphones Last Longer With Code Offload

Problem formulation: Battery is a scarce resource and will probably remain so for a longer time (no breakthrough in battery technology in sight). Remote execution can reduce energy consumption but leaves a number of challenges. What kind of computations should be off-loaded? How can we dynamically decide what to offload (according to network conditions, execution time etc.)? Important observations: off-loading of code works much better when the latency (RTT) is small.
Contributions:

- The authors introduce a framework to reduce programmer effort and to make cloud off-loading easy. MAUI (Mobile Assistance Using Infrastructure) combines extensive profiling with an ILP (integer linear programming) solver in order to make dynamic off-loading decisions and to optimize for energy reduction.
- Simple language keyword `[Remoteable]` allows (non-UI) computations to be off-loaded whenever this is beneficial
- Allows for easy partitioning of programs
- MAUI decides at runtime whether to invoke the local or remote method
- The ILP solves the optimization problem to determine whether to off-load based on the call graph (by doing global program analysis)
- MAUI enables developers to bypass the resource limitations of handheld devices and provides a lower barrier of entry
- Reduces the energy consumption by an order of magnitude while being able to adapt to changing network conditions and changing CPU demands

7.1 Paper: EagleEye: Wearable Camera-based Person Identification in Crowded Urban Spaces

Problem formulation: It can take a long time to identify people in crowds and especially for parents looking for their lost child this time can be critical. Furthermore, even state-of-the-art deep learning models often fail to identify low-resolution faces (i.e. from far away) accurately.
Contributions:

- The authors propose EagleEye, a wearable camera-based system that can help identify missing persons in large and crowded spaces

- This wearable device can help to quickly identify the missing person in the user's field of view in (soft) real time @ 1 fps
- They overcome several technical challenges in the design of this application:
  - As low-resolution faces can be hard to identify, the authors introduce *Identity Clarification Networks* into their pipeline to enhance facial features in the detected faces before the facial recognition layers. Instead of generating the most pleasing images (like SR-GANs usually do) ICN focuses on enhancing identity information for higher recognition accuracy.
  - To enable their solution to run on constrained mobile devices, the authors introduce a Multi-DNN execution pipeline based on *Content-Adaptive Parallel Execution* which prunes the search tree by excluding regions of the image that do not contain any faces from the super resolution and recognition layers.
  - To speed up the computation further, they divide the image into regions for which they are then able to parallelize the computations
  - Any heavy computations are off-loaded to the cloud, i.e. low resolution or non-frontal faces are off-loaded while frontal views are classified using a lightweight on-device recognition model

7.2 Paper: Edge Assisted Real-time Object Detection for Mobile Augmented Reality

Problem formulation: High accuracy object detection on Augmented Reality (AR) and Mixed Reality (MR) systems can be enabled with CNNs, however it is difficult to execute large networks on mobile devices. Off-loading object detection to the cloud can also be challenging as this introduces additional latency and lowers detection accuracy as the user's field-of-view might change while the data is being transferred.
Contributions:

- The authors propose a system which employs low-latency off-loading techniques, decouples the rendering pipeline from the off-loading pipeline and uses a fast object tracking method to maintain detection accuracy at the same time
- Dynamic ROI encoding, based on traditional region of interest (ROI) encoding:
  1. Divide each frame into *macroblocks*
  2. Adjust the encoding quality using a *quantization parameter* (QP)
  3. Use ROIs from last frame to determine the encoding quality on the current frame. Use low minimum confidence threshold for ROIs from region proposal network as not to miss new objects.

4. Calculate QP map based on calculated ROIs
5. Encode current frame according to the QP map

- Parallel streaming and inference:
  - Reducing inference and streaming latency is important and parallelizing the pipeline can reduce latency
  - Dependency-aware inference effectively pipelines and executes streaming and inference
  - Cut whole frame into 4 equisized slices
- Motion vector-based object tracking:
  - Utilize specialized hardware for H.264 and H.265 to compute motion vectors (normally used for video compression)
  - Use last cached frame and current frame, extract all motion vectors and infer where the bounding box has moved. Simply shift bounding boxes from previous frame and obtain new bounding boxes.
- Adaptive off-loading:
  - Determine off-loading based on two principles:
    - Eligible to off-load if previous off-loaded frame has been completely received by the edge cloud (network congestion)
    - Considered for off-loading if current frame differs significantly from the last off-loaded frame, i.e. off-load iff enough changes occur in order to reduce communication and computing cost (large motion of user or considerable change in pixels)
- In the experiments the authors conduct the device is connected to the server through a WiFi connection
- The system manages to reduce latency drastically and runs at 60fps while increasing the accuracy $20.2 \sim 34.8\%$ and consumes very few resources thus enabling the actual rendering process to utilize more resources