Wearables (2)

CSE 162 – Mobile Computing
Hua Huang
Department of Computer Science and Engineering
University of California, Merced

In this lecture

- Energy conservation techniques in Android
- Risq: a wearable smoking detection system

- Battery energy is most constraining resource on mobile device
- The problem is even more acute in the wearable devices, where the sizes are even smaller.
- Most resources (CPU, RAM, WiFi speed, etc) increasing exponentially except battery energy

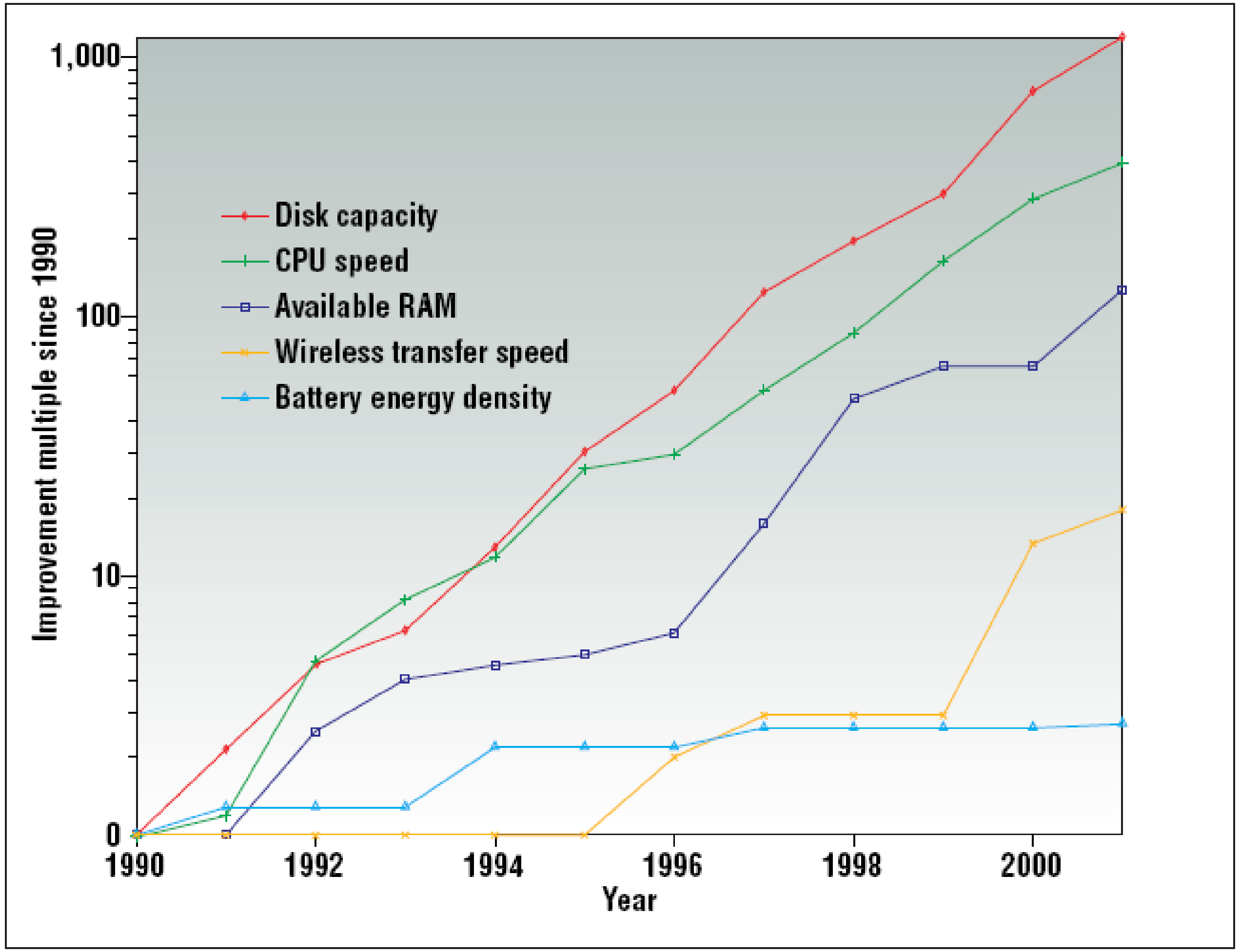


Figure 1. Improvements in laptop technology from 1990–2001.

Battery Conservation in mobile devices

- Three things to help
  - Make apps *Lazy First*
  - Take advantage of platform features
  - Use tools to identify battery draining components

Lazy First

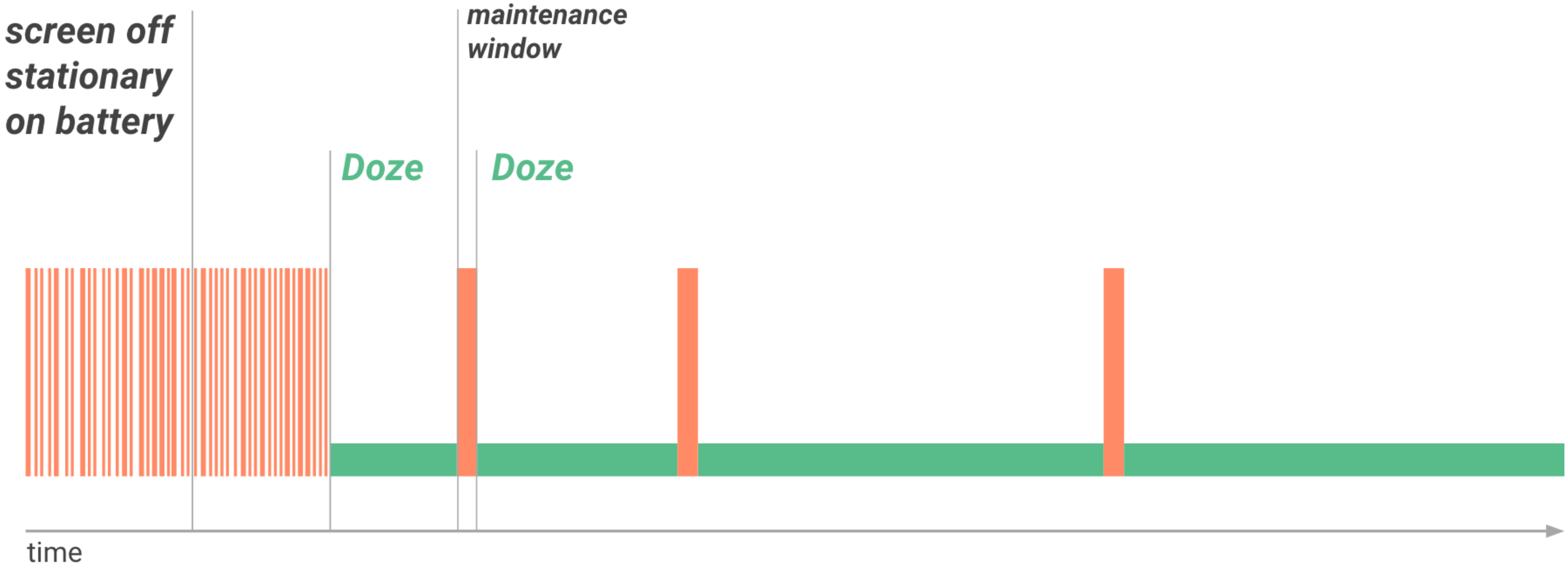
- **Reduce**
  - Are there redundant operations your app can cut out?
- **Defer**
  - Does an app need to perform an action right away?
- **Coalesce**
  - Can work be batched, instead of putting the device into an active state many times?

Android platform features

- **Doze** and **App Standby**:
  - Two power saving features
  - Manage how apps behave when a device is not connected to a power source
- Doze
  - Defers background CPU and network activities when the device is unused for a long time
- App Standby
  - Defers network activity when user has not recently interacted

Doze Mode

- Starts when a device is **unplugged** and **stationary** for **a period of time**, with the **screen off**
- Periodically, the system exits Doze to let apps complete their deferred activities, such as running pending syncs, jobs, alarms, and network access

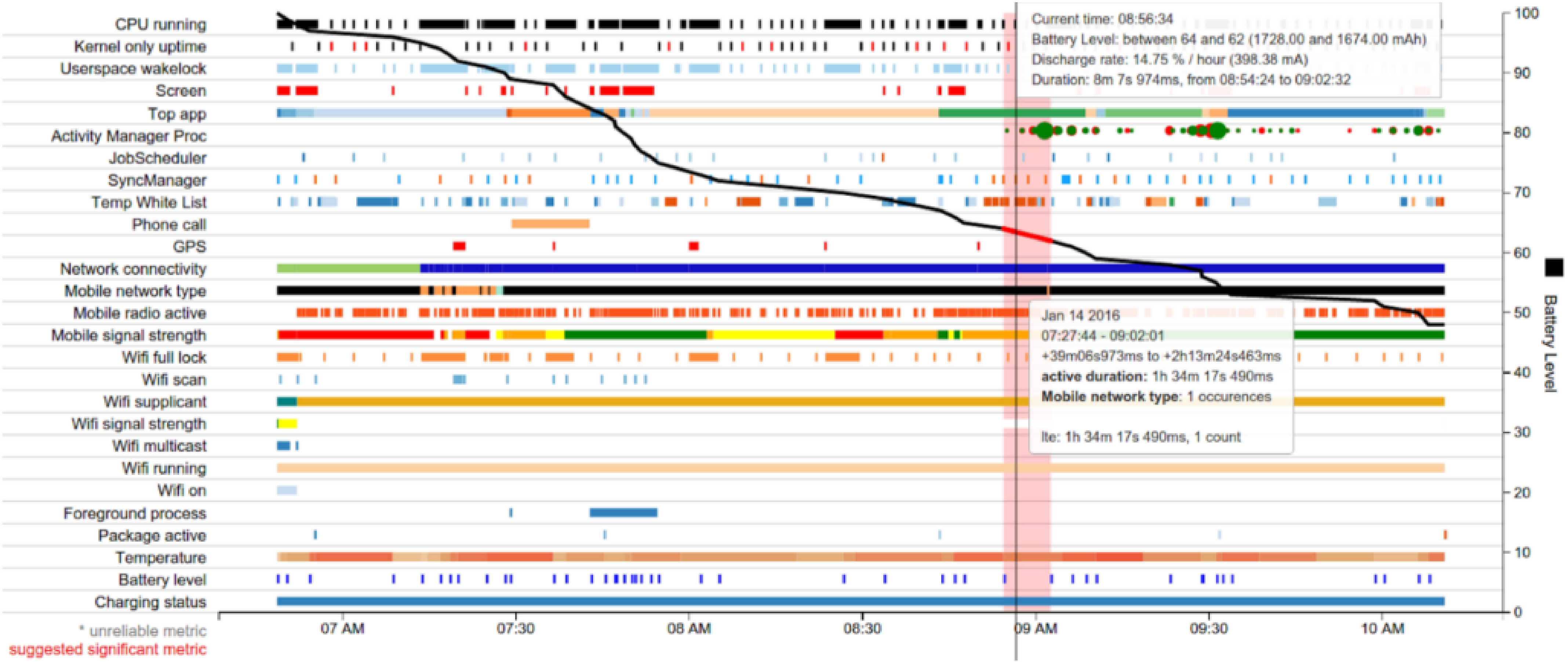


Android App Standby

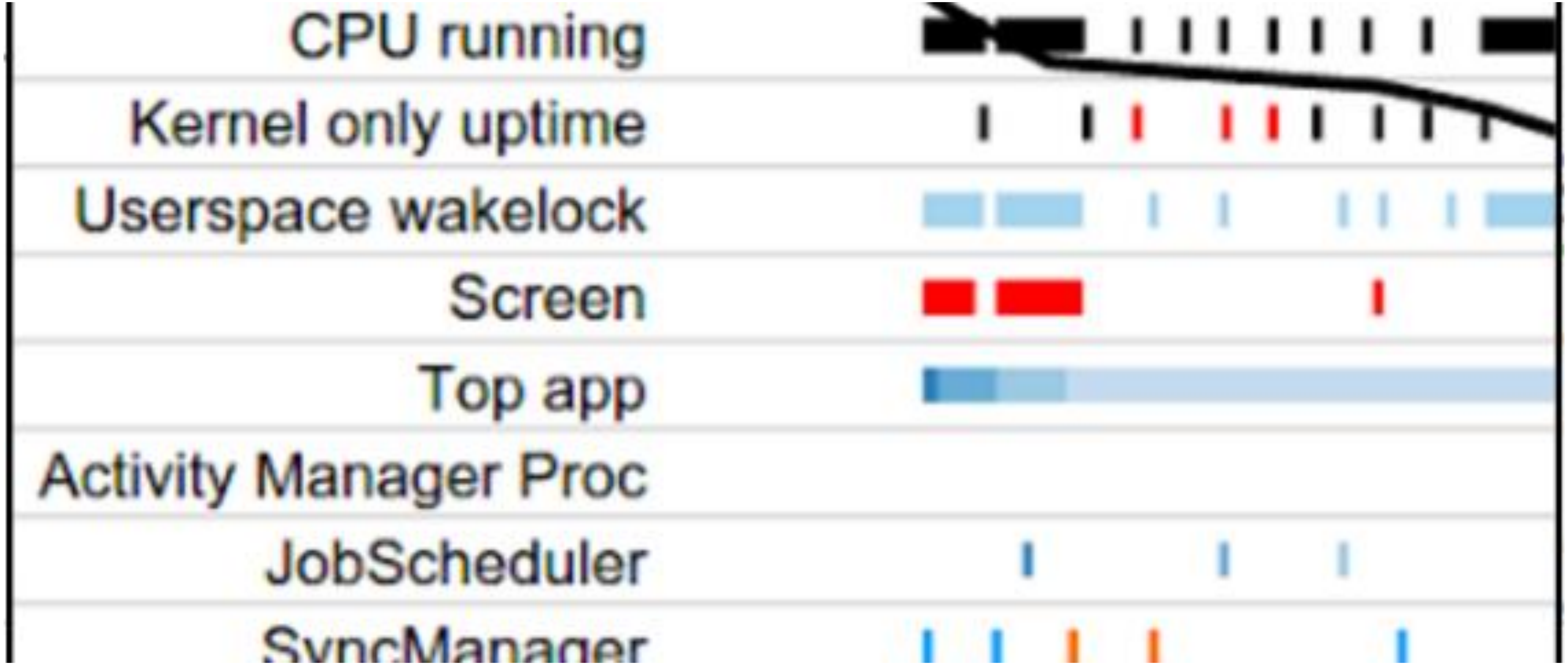
- The system puts an app into Standby mode and defers its network access if for a certain period of time,
  - the user has not explicitly launched the app, and
  - the app doesn’t have a foreground process (activity), and
  - the app doesn’t generate notifications, and
  - the app is not an active device admin app.
- When the phone is plugged to power, apps are released from standby state
- If the device is idle for a long period of time, idle apps access network around once a day

Battery Historian: A tool to analyze power use in Android

provides a system-wide visualization of various app and system behaviors, along with their correlation against battery consumption over time.



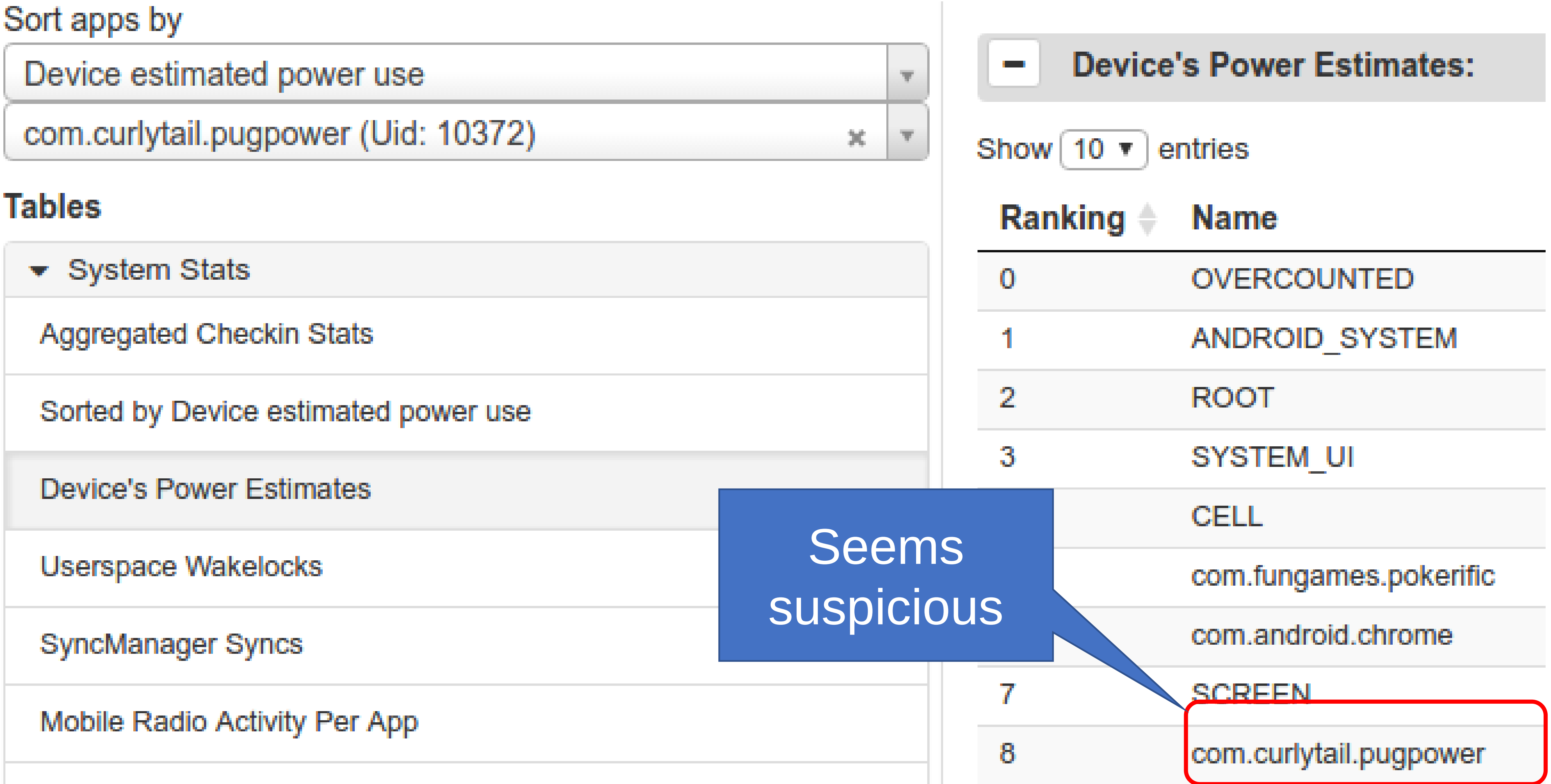
A closeup look



Battery historian: App-specific view

- Provides tables for the following data:
  - The app’s estimated power use on the device.
  - Network information.
  - Wakelocks.
  - Services.
  - Process info.

Display the ranking of app powers



App specific data

System Stats | History Stats | App Stats

Copy

Application	com.curlytail.pugpower
Version Code	117
UID	10372
Device estimated power use	3.66%
Total number of wakeup alarms	0

+ Network Information:

− Wakelocks:

Show 5 entries

Search: Copy

Wakelock Name	Full Time	Full Count	Partial Time	Partial Count	Window Time	Window Count
com.curlytail.treatsIntentService		0	1h 4m 34s 323ms	7		0
alarm		0	177ms	8		0

Showing 1 to 2 of 2 entries

Previous 1 Next

Other cases where Battery Historian can help

- Examples:
  - Firing wakeup alarms overly frequently (every 10 seconds or less).
  - Continuously holding a GPS lock.
  - Scheduling jobs every 30 seconds or less.
  - Scheduling syncs every 30 seconds or less.
  - Using the cellular radio more frequently than you expect.

Recognizing Smoking Gestures with Wearable Inertial Measurements Unit (IMU)

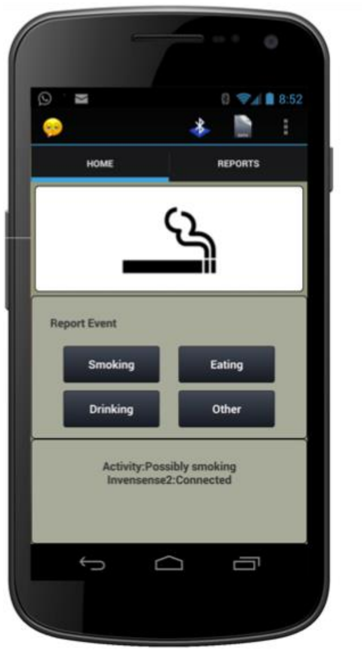
Smoking

- According to CDC, smoking is responsible for
  - 440,000 deaths in the United States
  - $96 billion in medical costs
  - $97 billion in lost productivity
- Over a billion smokers worldwide!

Smoking Cessation

- 40% smokers try to quit each year.
- Most efforts end in relapse.
- Well-timed interventions help!
  - Less than 10 % success rate
  - Requires presence of a ubiquitous agent

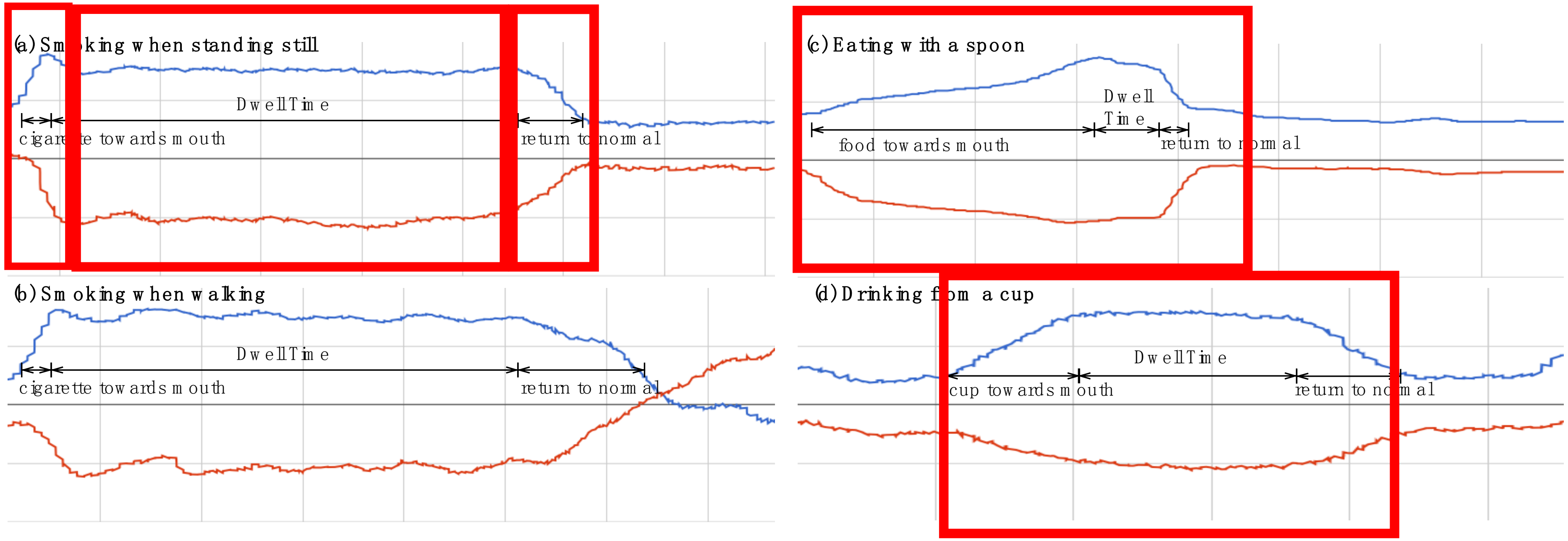
RisQ: A Mobile Solution for Intervention



- Smartphone
  - Always with the user
  - Can sense user environment
  - Real-time intervention

- Wristband
  - Equipped with 9-axis Inertial Measurement Unit (IMU)
  - Real-time smoking detection

Hand-to-mouth gesture characteristics

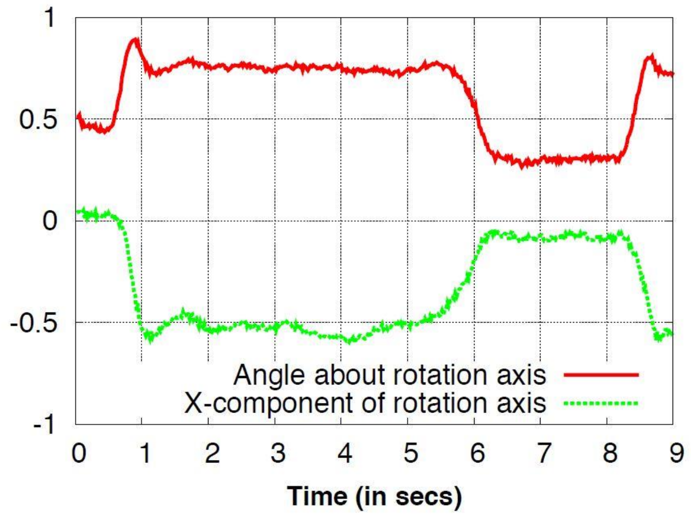


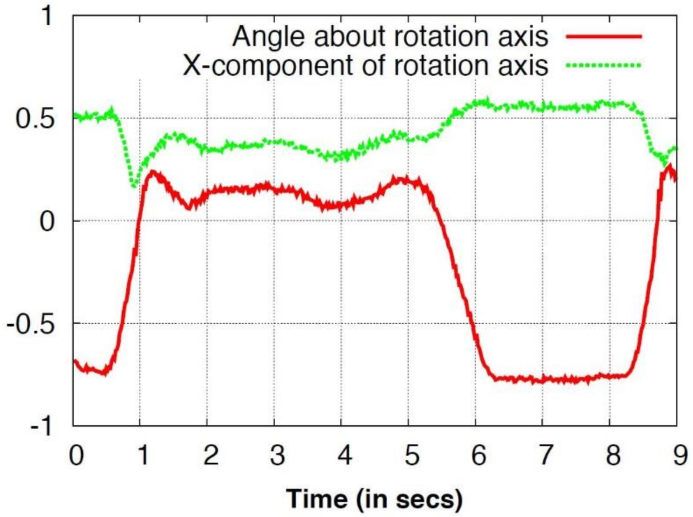
IMU Signals for various hand-to-mouth gestures

Outline

- Introduction
- Challenges
- Data Collection using IMUs
- Data Processing Pipeline
- Evaluation
- Conclusion

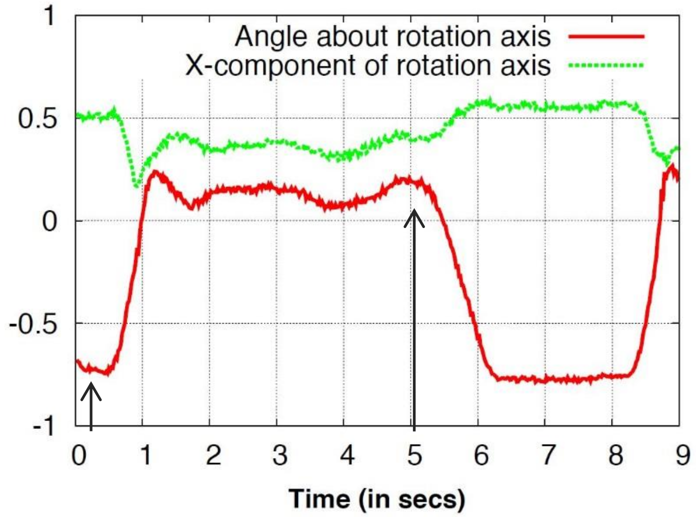
1: Orientation-dependent Characteristics





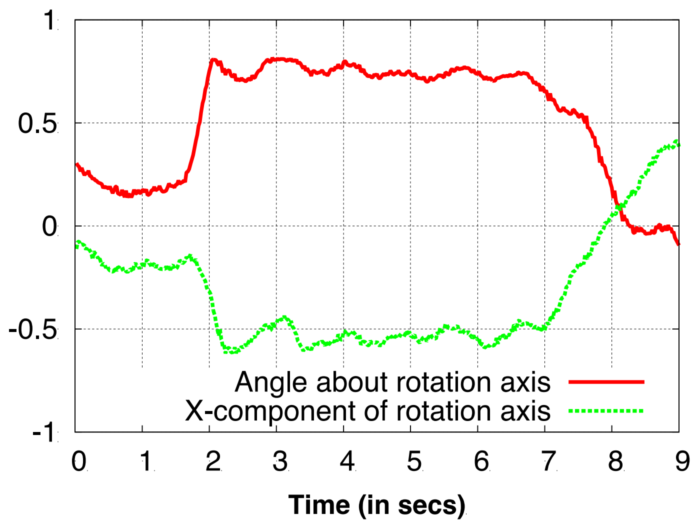
Signal characteristics change with user's body orientation

2: Unknown Gesture Boundaries



How to identify gesture boundaries in a passive manner?

3: Collecting labels for training



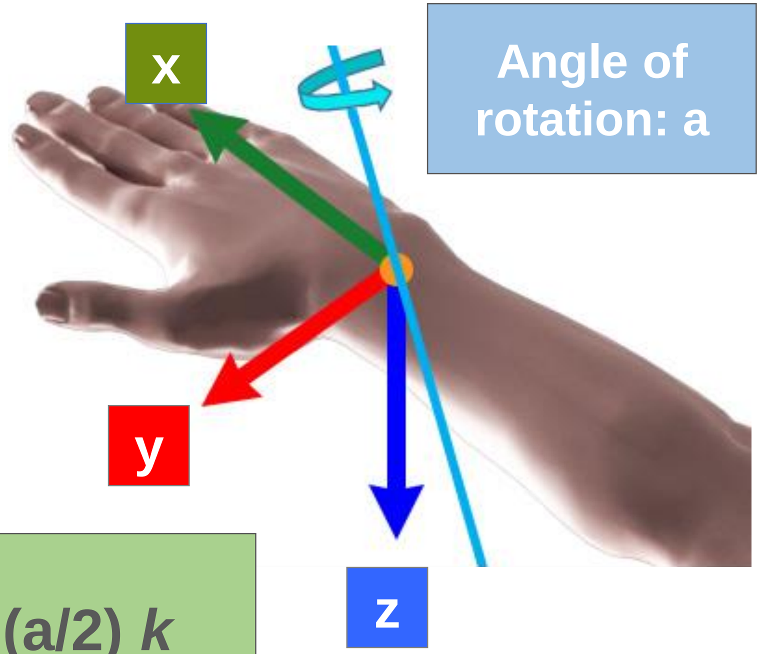
How to collect fine-grained labels for training a classification model?

Outline

- Introduction
- Challenges
- Data Collection using IMUs
- Data Processing Pipeline
- Evaluation
- Conclusion

Quaternion recap

- Mathematical entity to represent orientation of an object in 3D space
- **q** = **q_s** + **q_x** *i*+**q_y** *j*+**q_z** *k*
- One scalar and 3 imaginary components



q = cos(a/2) + x sin(a/2) *i* + y sin(a/2) *j* + z sin(a/2) *k*

3D coordinates using Quaternions

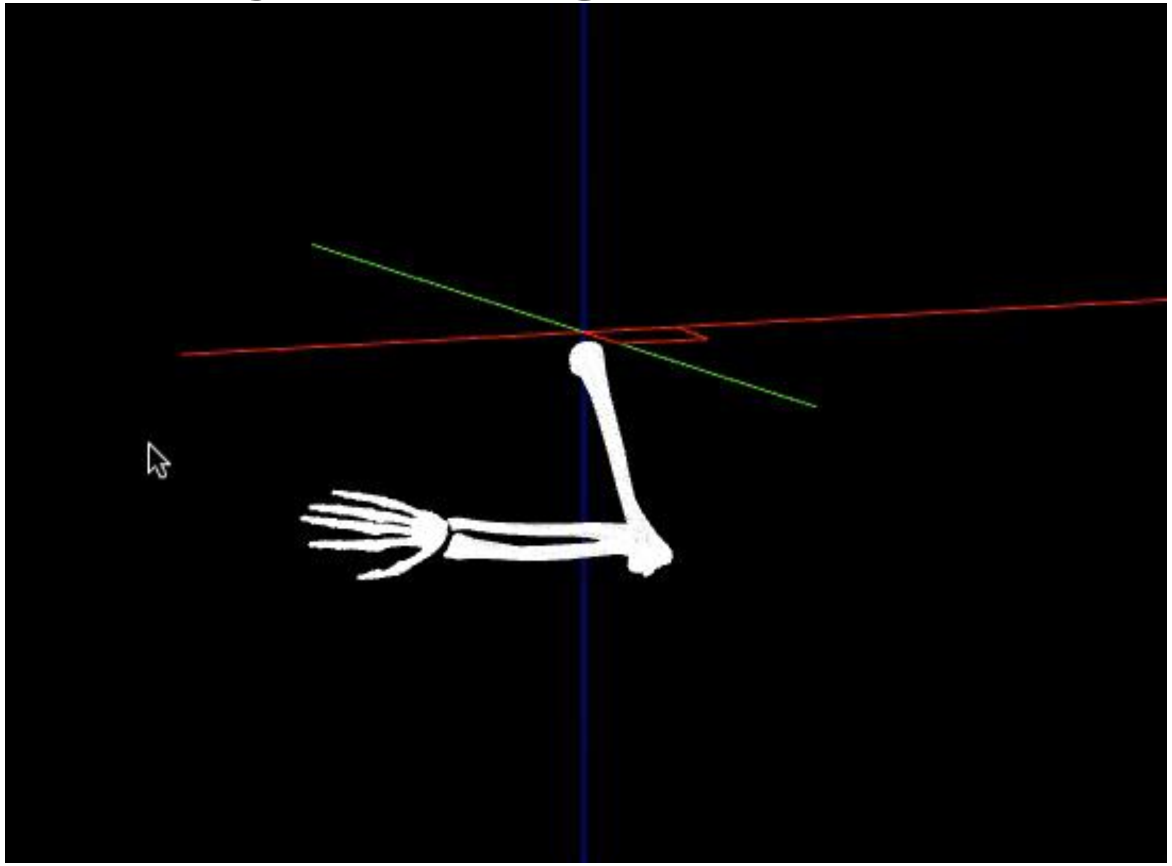
- Point **p** w.r.t. IMU's local frame of reference
- IMU device orientation in the form of a quaternion **q**
- Coordinates of **p** w.r.t world frame of reference

p' = q.p.q'

$$q = \cos(\mathbf{a}/2) + \mathbf{x}\sin(\mathbf{a}/2)\, i + \mathbf{y}\sin(\mathbf{a}/2)\, j + \mathbf{z}\sin(\mathbf{a}/2)\, k$$

$$q' = \cos(\mathbf{a}/2) - \mathbf{x}\sin(\mathbf{a}/2)\, I - \mathbf{y}\sin(\mathbf{a}/2)\, j - \mathbf{z}\sin(\mathbf{a}/2)\, k$$

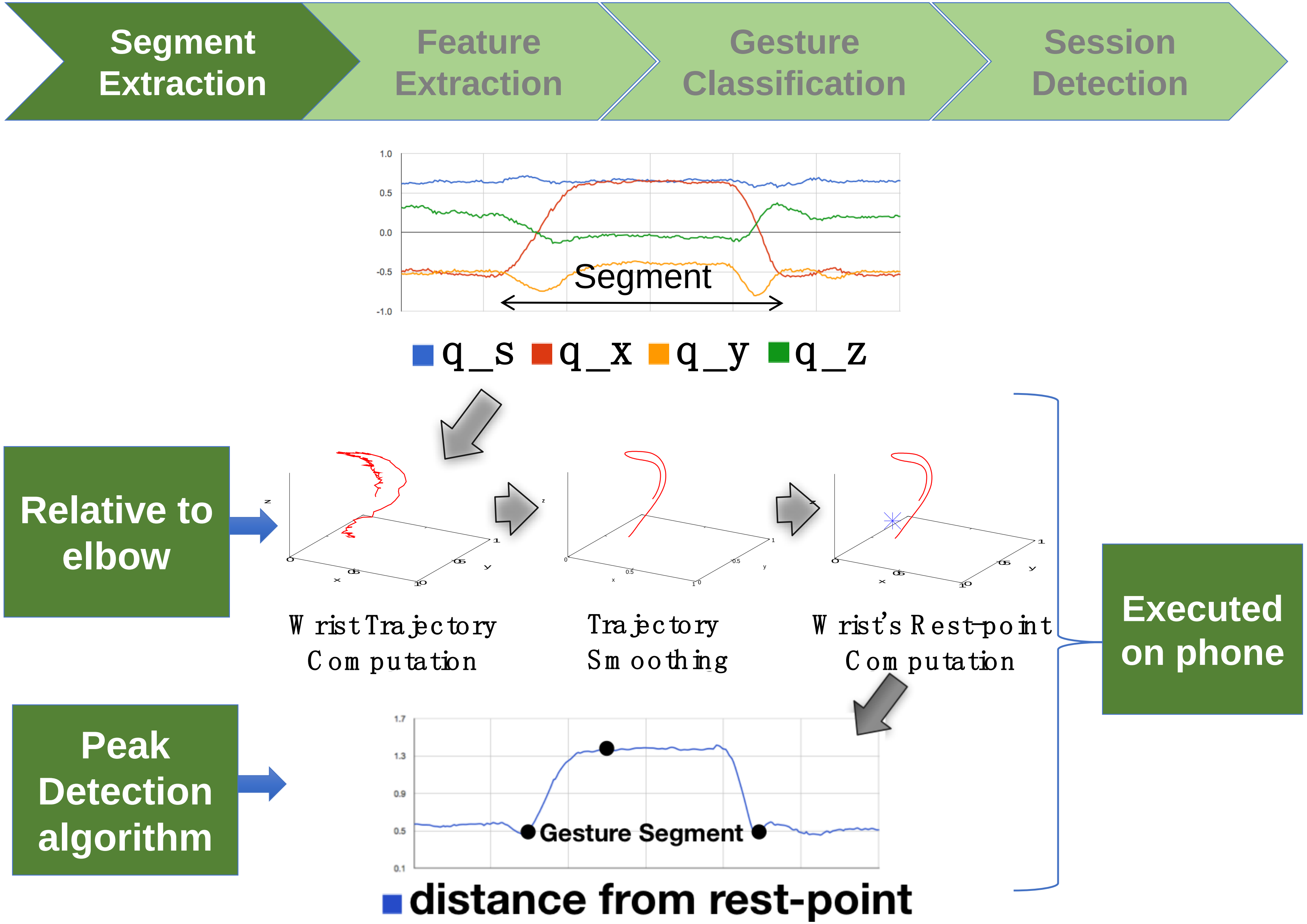
Wrist Trajectory using Quaternions



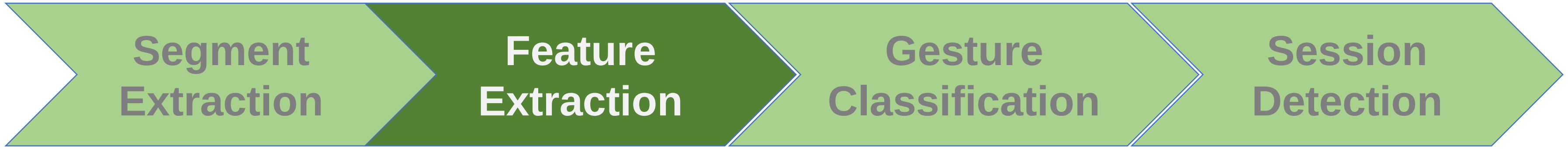
Visualizing gestures using a wristband and an armband equipped with IMUs

Outline

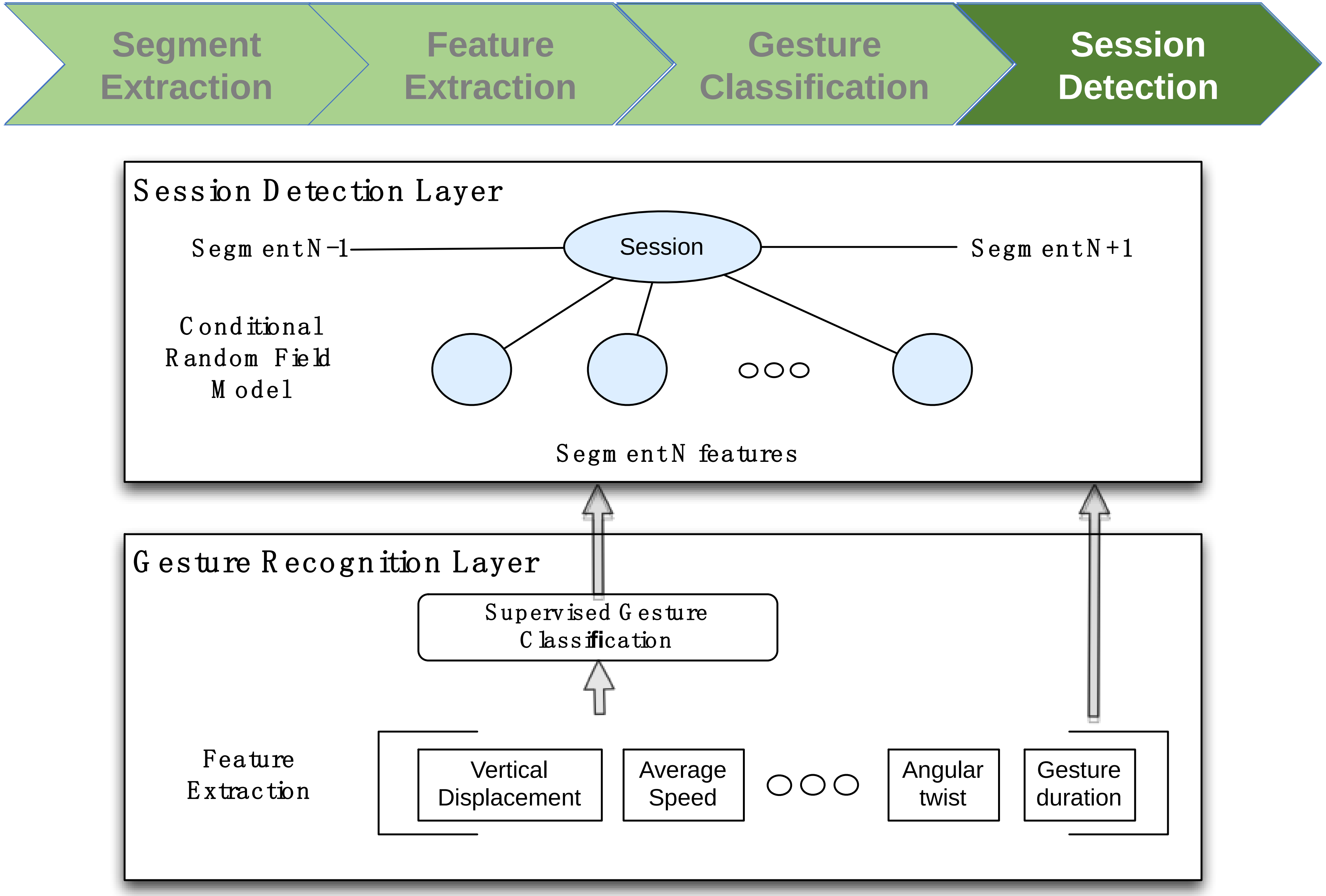
- Introduction
- Challenges
- Data Collection using IMUs
- Data Processing Pipeline
- Evaluation
- Conclusion



Segment Extraction
Feature Extraction
Gesture Classification
Session Detection
1.0
0.5
0.0
-0.5
-1.0
Segment
q_s q_x q_y q_z
Relative to elbow
Wrist Trajectory Computation
Trajectory Smoothing
Wrist's Rest-point Computation
Executed on phone
Peak Detection algorithm
1.7
1.3
0.9
0.5
0.1
Gesture Segment
distance from rest-point



- Orientation Independent Features
  - A set of 34 spatio-temporal features
- Duration-based features (4)
  - Gesture duration, time to raise arm, etc.
- Velocity-based features (6)
  - Maximum wrist speed, etc.
- Displacement-based features (6)
  - Vertical displacement, XY displacement, etc.
- Angle-based features (18)
  - Angle with the gravity, angular velocity, etc.



Segment Extraction
Feature Extraction
Gesture Classification
Session Detection
Session Detection Layer
Segment N-1
Session
Segment N+1
Conditional Random Field Model
Segment N features
Gesture Recognition Layer
Supervised Gesture Classification
Feature Extraction
Vertical Displacement
Average Speed
Angular twist
Gesture duration

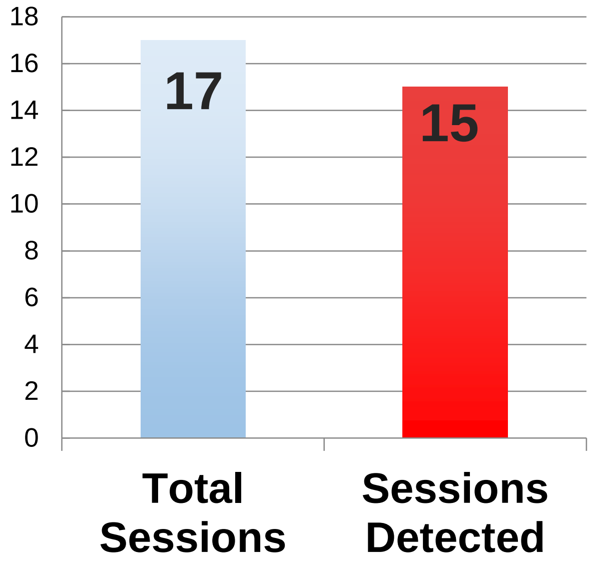
Outline

- Introduction
- Challenges
- Data Collection using IMUs
- Data Processing Pipeline
- Evaluation
- Conclusion

Evaluation Dataset

- Dataset
  - 28 hours of data from 15 volunteers
  - 17 smoking sessions (369 puffs)
  - 10 eating sessions (252 food bites)
  - 6 drinking sessions

Smoking Session Detection



Statistic	Avg ± Std Dev
Duration of smoking sessions	326.21 ± 19.65 s
Error in estimation	65.7 ± 30.6 s

Leave-one-session-out Cross-validation

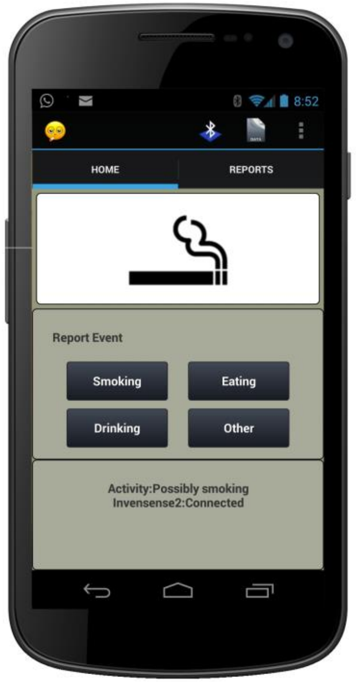
Smoking Gesture Recognition

Mechanism	Performance Metrics			
	Accuracy	Recall	Precision	FPR
Random Forests	93.00%	0.85	0.72	0.023
CRF	95.74%	0.81	0.91	0.005

- **10-fold Cross-validation**
  - 369 puffs
  - 252 bites
  - 4976 other gestures

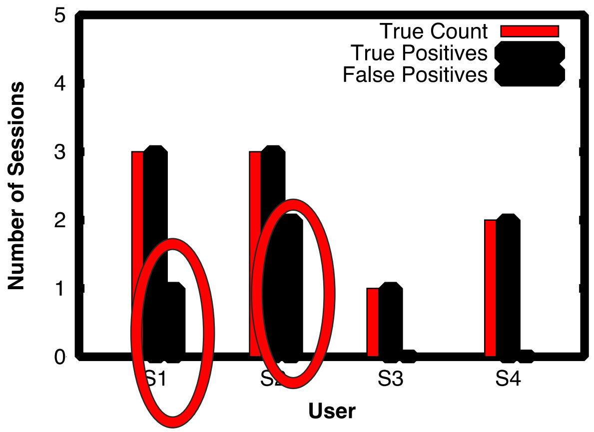
CRF improves precision at a cost of slight drop in recall

User Study

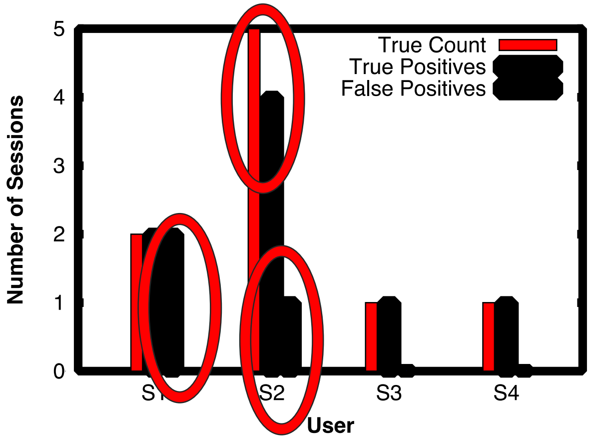


- Recruited 4 subjects for 3 days.
- Used our smoking detection app developed for Android OS.

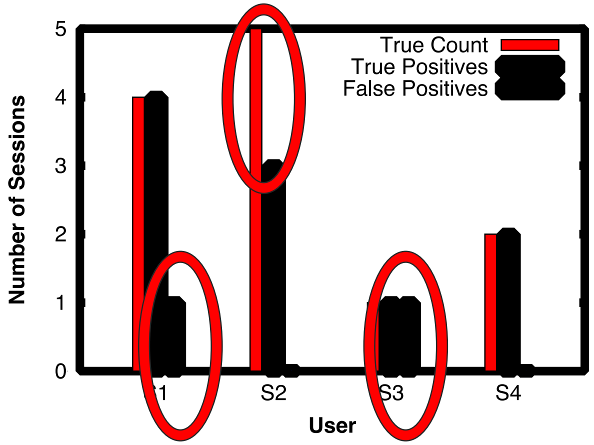
User Study



Day 1



Day 2



Day 3

Rarely missed any smoking session.

Eating Gesture Recognition

Mechanism	Eating Sessions		All data	
	Recall	**Precision**	**Recall**	**Precision**
Bite-Counter	0.60	0.57	0.65	0.03
Random Forests	0.92	0.78	0.69	0.64
CRF	N/A	N/A	0.64	0.78

- Eating gesture recognition
- Bite-Counter detects food bites when user explicitly indicates that eating session is in progress.

System Overhead

Statistic	**Value**
Time for segmentation	92.34ms
Time for feature extraction	79.88ms
Time for CRF inference	5.89ms
Memory	12-20MB
Binary Size	1.7MB

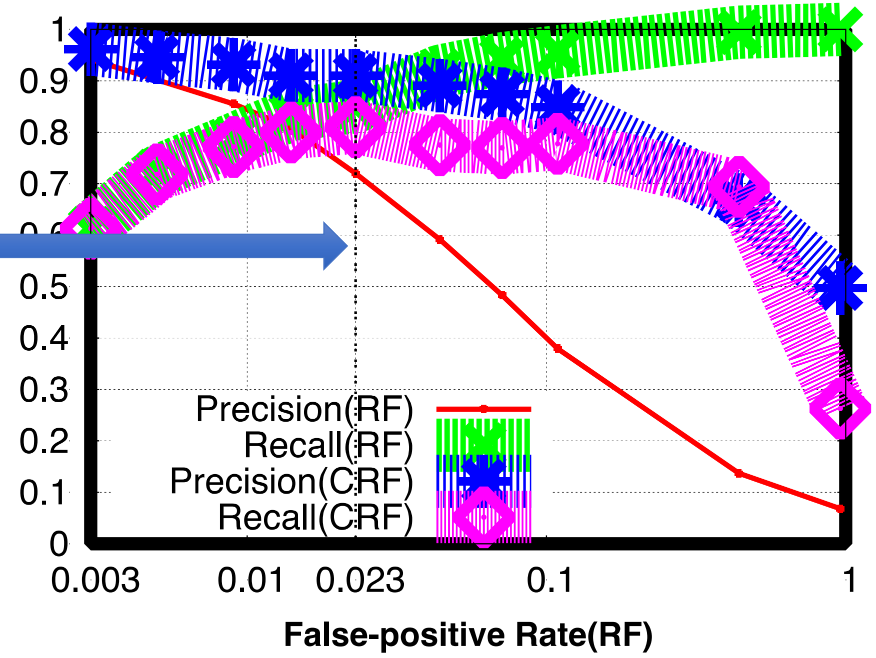
Measured on Samsung Galaxy Nexus

Conclusion

- An algorithm to recognize hand-gestures using a wristband
  - Demonstrated an application to detect smoking in real-time.
- Wearable accessories present a great platform to sense health-related behaviors like smoking, eating, and so on.
- Remarkable opportunity to create effective intervention strategies using wearables.

Optimizing Performance

Optimal cost for best performance



- Use a cost function during RF classifier training to assign penalty for missing a smoking gesture.
- High cost results in lower precision
- Low cost results in lower recall and low FPR