

# Activity Recognition & Condition-based Maintenance

- RecoFit: using a wearable sensor to find, recognize, and count repetitive exercises Dan Morris, T. Scott Saponas, Andrew Guillory, Ilya Kelner, ACM CHI 2014
- CNN-based sensor fusion techniques for multimodal human activity recognition, ISWC 2017
- Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition, Sensors 2016
- Vibration Analysis for IoT Enabled Predictive Maintenance, Deokwoo Jung, Zhenjie Zhang, Marianne Winslett, ICDE 2017

# RecoFit: using a wearable sensor to find, recognize, and count repetitive exercises

Dan Morris, T. Scott Saponas,  
Andrew Guillory, Ilya Kelner

ACM CHI 2014



Figure 3: Data collection and evaluation hardware.



Track exercises from an arm-worn sensor, with no user-specific training, and no intervention from the user during a workout

# RecoFit

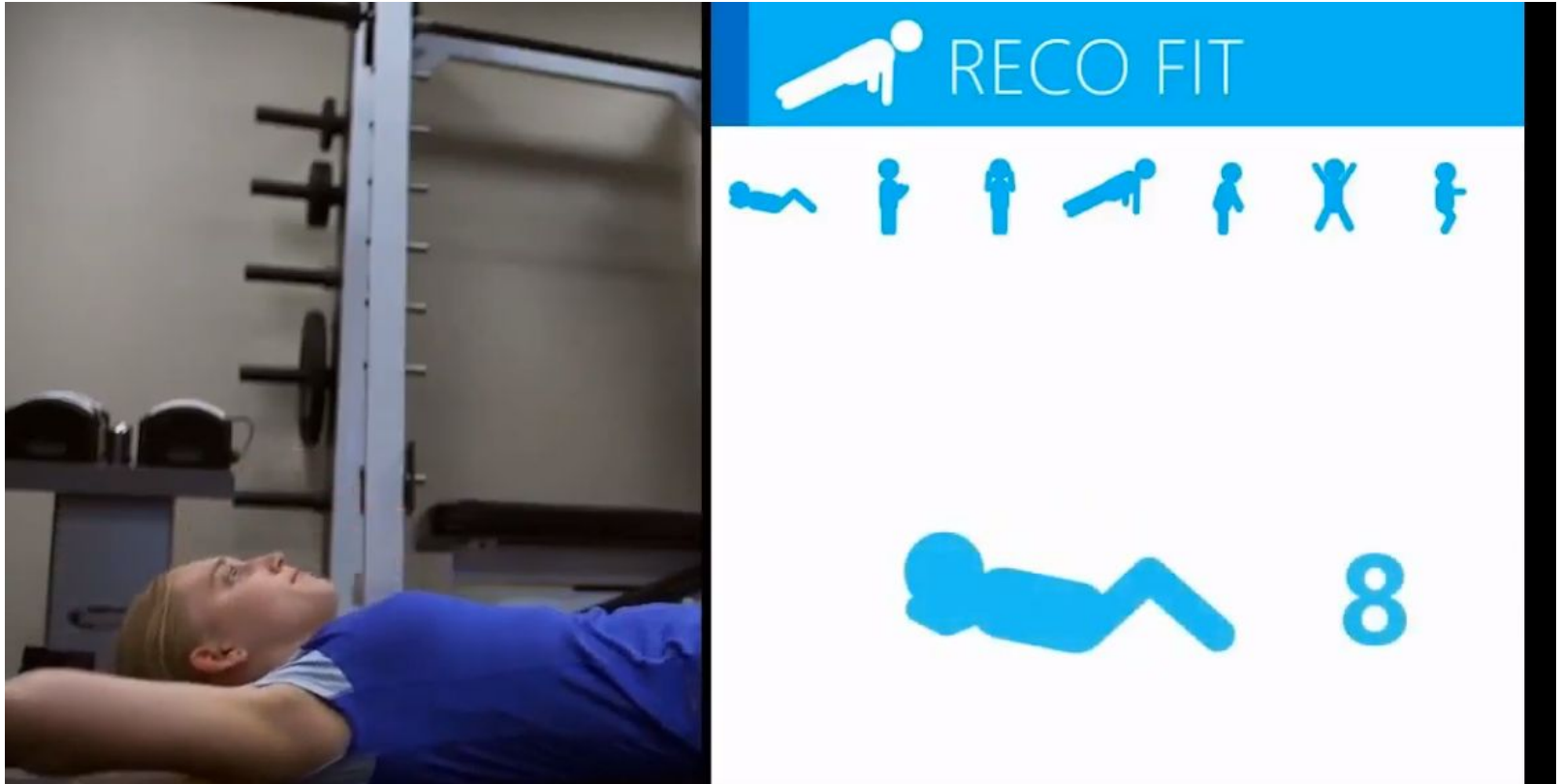
Automatically Finding, Recognizing,  
and Counting Repetitive Exercises

Dan Morris, Scott Saponas, Andrew Guillory, Ilya Kelner  
Microsoft Research

ACM CHI 2014

<https://www.youtube.com/watch?v=zpa4rVGIO68>

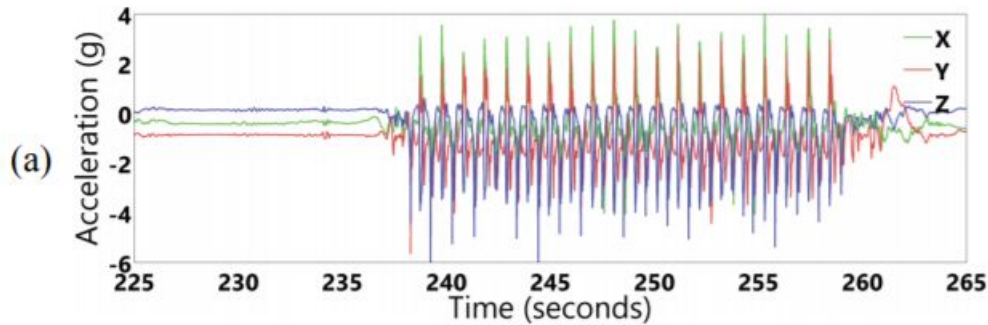
Track exercises from an arm-worn sensor, with no user-specific training, and no intervention from the user during a workout



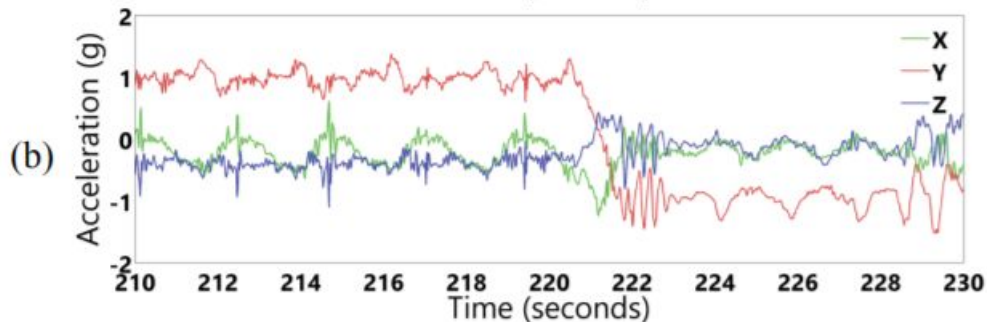
# Challenges

- Exercise looks very similar to non-exercise
  - Time spent actually exercising within the context of a workout session may be as little as 10% or as much as 100% of the total session
  - Between exercises, one walks around the workout space, socializes, stretches, rests, drinks water, selects and retrieves equipment, etc.
  - Good news: exercise is typically **more periodic** (i.e., repetitive) than non-exercise => autocorrelation
  - Challenges: how to differentiate exercise from walking (periodic) and deal with stretching activities?

# Challenges



**The (rare) easy case:** a participant goes from a relatively motionless state to rapid movement



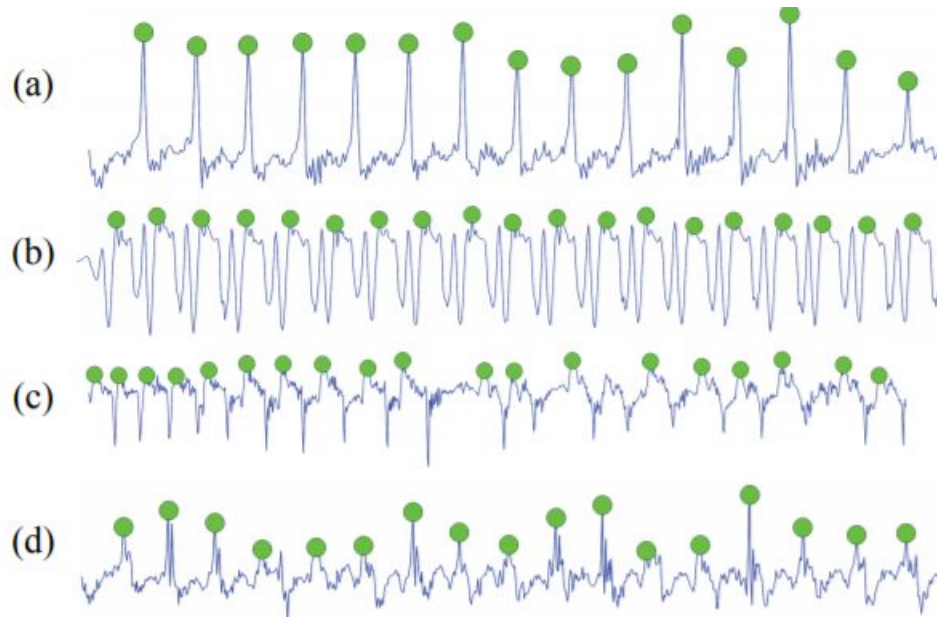
When a participant finishes a set, puts weights down, and stretches a bit; exercise and non-exercise amplitude are almost identical

# Challenges

- Variability in form
  - No user-specific training of our system, variability in users' interpretations of exercise descriptions, and their ability to consistently execute a particular form => this has a tremendous impact on recognition accuracy
  - Assume end-users would have some familiarity with the available exercises, but that users would not typically watch a proscriptive video or have access to a coach who would refine their form
  - Difficult to address the problem of variation in form other than large-scale training data collection with enough flexibility to elicit such variation

# Challenges

- Temporal Irregularities



**Figure 2: Counting challenges. (a) Consistent large peaks. (b) Each repetition contains multiple similar peaks. (c) Irregular timing and form. (d) Irregular amplitude. Circles indicate repetitions, correctly identified by our algorithm.**

**Jumping jack:** a simple peak-picking or zero-crossing approach will yield an accurate count  
A typical set of **squats**, with two bursts of acceleration per repetition, resulting in a “double peak” for each count



# Challenges

- Unpredictable device orientation
  - With an arm- or wrist-worn inertial sensor, the form factor itself presents a challenge
  - In real-world use, an arm-worn sensor will naturally rotate differently around different users' arms, as a consequence of preference and natural fit.
  - Trust the axis pointing along the arm (a watch, for example, always has its face pointed out, in a readable orientation), but that the device might **rotate** arbitrarily around the arm

# Data Collection

- SparkFun “Razor IMU” inertial sensor, which includes a 3-axis accelerometer and a 3-axis gyroscope
- Armband included a battery and a Bluetooth radio, which transmitted sensor values to a PC at 50Hz



# Data Collection: Lab Setting

- Importance of encouraging natural variability in training data
- Anecdotally, this project was begun with a smaller data set (30 participants), collected in a space-constrained laboratory environment that did not aesthetically resemble a gym, with clear instructions regarding sequencing and form
- **Cross-validation** on this data set was unrealistically encouraging: segmentation, recognition, and counting were all close to perfect
- This data was used to **initially to train** our system and deployed a real-time prototype in a more realistic environment, and it quickly became clear that early success was the result of **“robotic” behavior** among training participants
- Segmentation dropped to precision/recall levels close to **50% when users were in a more natural environment**

# Data Collection: Naturalistic

- “Natural” Environment and Procedure Design
  - 94 participants (28 female), ages 18-58 ( $\mu=34.2$ )
  - 126 sessions of training data; Sessions averaged 38min, about half of which was “non-exercise” time

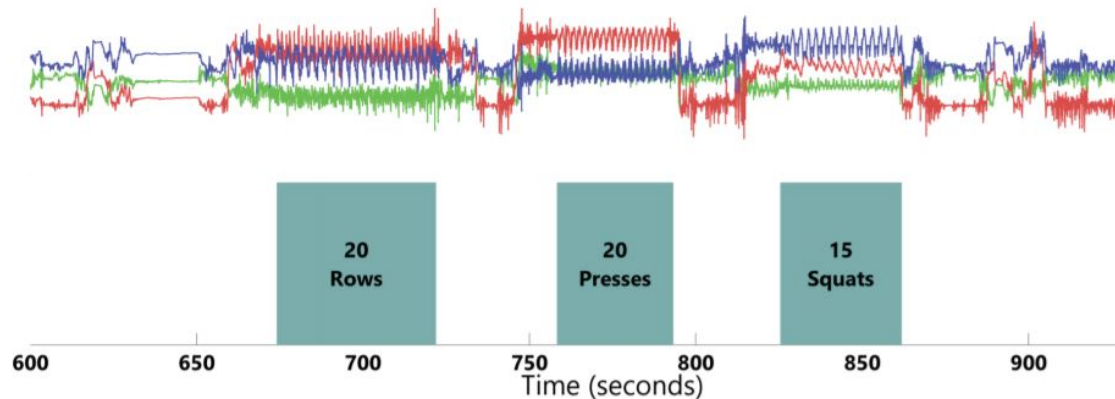
<b>Study A (4-class)</b>	Squat, crunch, pushup, shoulder press
<b>Study B (4-class)</b>	Curl, jumping jack, triceps extension, dumbbell row
<b>Medium (7-class)</b>	Squat, curl, crunch, pushup, triceps extension, dumbbell row, jumping jack
<b>Large (13-class)</b>	Crunch, row, punch, jumping jack, kettlebell swing, triceps extension, pushup, rowing machine, Russian twist, back fly, shoulder press, squat, curl

# Major Challenges

- **Segmenting** exercise from non-exercise
- Recognizing **which exercise** is being performed
- **Counting** repetition

# Segmenting exercise from non-exercise

- Step 1: reprocessing
- Step 2: feature computation
- Step 3: classification
- Step 4: aggregation



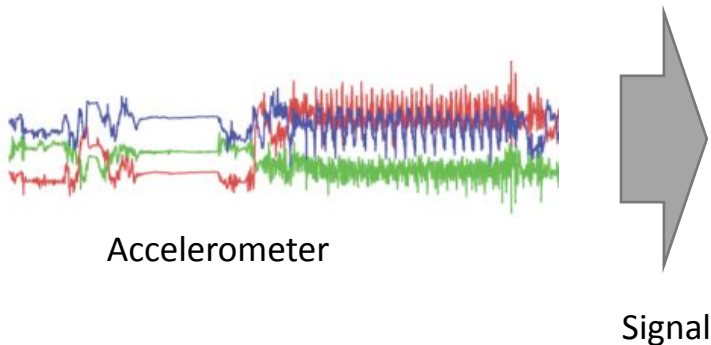
**Figure 4: Segmentation overview. The segmenter infers exercise periods (bottom) from accelerometer data (top).**

# Segmenting exercise from non-exercise

- Step 1: reprocessing
  - Accelerometer and gyroscope data are **smoothed** with a **Butterworth low-pass filter** (-60dB at 20Hz), then windowed into **5-second windows sliding at 200ms** (i.e., each 5s window shares 4.8s of data with the previous window)

# Segmenting exercise from non-exercise

- Step 2: feature computation
  - Each 5-second window is transformed into **224 features** that we use to characterize exercise
  - 28 features computed over each of 8 one-dimensional “signals” (four from accel + four from gyro)

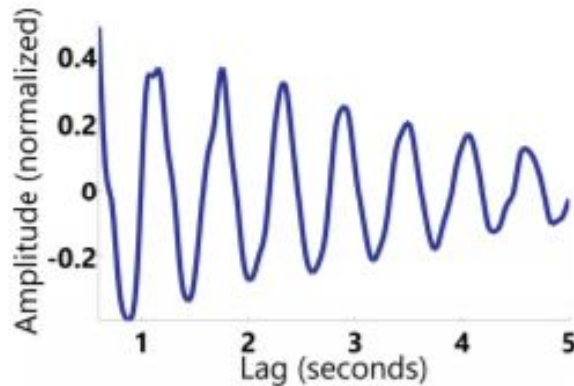


- 1) **aX**: the X-axis accelerometer signal
- 2) **aXmag**: the magnitude of the accelerometer signal at each sample, i.e.  $\sqrt{a_x^2 + a_y^2 + a_z^2}$ .
- 3) **aPC1**: the projection of the three-dimensional accelerometer signal onto its first principal component. This is the movement along the axis that demonstrates the most variance within this window, or – anecdotally – the most “interesting” rotation of the window.
- 4) **aYZPC1**: the projection of *only* the Y and Z axes onto the first principal component of those two axes. This captures *movement perpendicular to the arm*, which allows us to derive information from the Y and Z axes despite the unknown rotation of the armband.

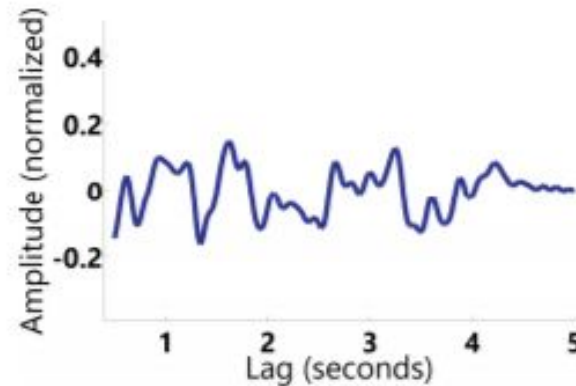


# Segmenting exercise from non-exercise

- Step 2: feature computation
  - Intuition: exercise is usually **more repetitive** than non-exercise
  - Compute segmentation features based on each signal by leveraging **autocorrelation**, i.e. the cross-correlation of a signal with itself
    - If a signal has a periodic component at frequency  $f$ , a peak in the autocorrelation at lag  $1/f$  will appear
    - Autocorrelation of white noise shows no peaks at all (no periodicity)



**Autocorrelation of repetitive activity**  
is smooth with prominent peaks



**Autocorrelation of non-repetitive activity** is  
jagged with weak peaks

# Segmenting exercise from non-exercise

- Step 2: feature computation
  - Compute the autocorrelation of each signal, normalize it to have value 1 at zero lag, exclude lags less than 0.5s, and compute the following five features
    - **Number of autocorrelation peaks** (note that this feature does not vary monotonically with repetitiveness; too many peaks is indicative of irregular movement, but too few is also consistent with a noisy signal)
    - **Prominent peaks:** The number of autocorrelation peaks that are larger than their neighboring peaks by a threshold and are more than a threshold lag away from their neighboring peaks. **In general, more prominent peaks are present in exercise than non-exercise**
    - **Weak peaks:** The number of autocorrelation peaks that are within a threshold height of their neighboring peaks and are less than a threshold lag away from their neighboring peaks. **In general, more weak peaks are present in non-exercise than exercise**
    - **Maximum autocorrelation value:** In general, **a higher maximum value** (other than the initial peak at zero lag, which we do not consider) is consistent with **exercise**
    - Height of the first autocorrelation peak after a zero-crossing

# Segmenting exercise from non-exercise

- Step 2: feature computation
  - To improve the precision of segmentation boundaries, compute the RMS, mean, standard deviation, and variance for the first half and second half of each window
    - RMS: The root-mean-square amplitude of the signal. The intuition here is that faster motion is more likely to correspond to exercise than non-exercise
    - Power bands: The magnitude of the power spectrum in 10 bands spaced linearly from 0.1-25Hz (10 features)
    - Mean, standard deviation, and variance (3 features)
    - Integrated RMS: The root-mean-square amplitude of the signal after cumulative summation (which roughly takes us from “acceleration” space to “velocity” space)

# Segmenting exercise from non-exercise

- Step 3: classification
  - Every 5-second window (200ms step) in our training data is labeled “exercise” or “non-exercise” from the ground truth information (walking is considered “non-exercise”)
  - For each window, compute these 224 features, and the resulting feature matrix is used to train an L2 linear support vector machine (SVM), which predicts either “exercise” or “non-exercise” for each 5-second window

# Segmenting exercise from non-exercise

- Step 4: aggregation w/ accumulator strategy
  - Possible to see disagreements with the actual exercise/non-exercise state for short periods, e.g. for **brief dynamic stretches during non-exercise**, or **pauses in the middle of an exercise**
  - **Accumulator strategy** for aggregation:
    - Start in the non-exercise state
    - Every time a prediction of “exercise”: increment a value (the accumulator); otherwise, a prediction of non-exercise: decrement the accumulator
    - When the accumulator reaches a threshold, the system enters the “exercise” state. **This threshold is set to the equivalent of 6 seconds.**
    - Maintain a history buffer, so that when the system enters the “exercise” state, it can provide the entire window of exercise to the recognition and counting systems.
    - Cf) this process is simply performed in reverse to detect the end of an exercise

# Recognizing which exercise is being performed

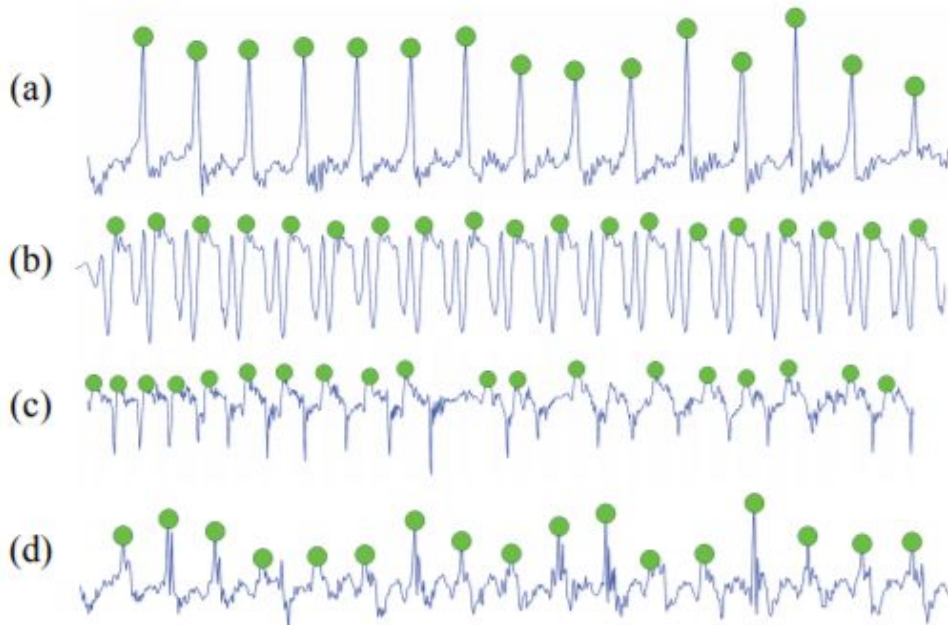
- Step 1 Preprocessing
- Step 2 Feature Computation
  - 20 features computed over each of 3 signals (i.e., aX, aYZPC1, and gPC1 ) derived
    - Autocorrelation bins: 5 evenly-spaced bins of the 5- second autocorrelation – summed per bin (5 features)
    - RMS: The root-mean-square amplitude of the signal.
    - Power bands: The magnitude of the power spectrum in 10 bands spaced linearly from 0.1-25Hz (10 features)
    - Mean, standard deviation, kurtosis, interquartile range (4 features).
- Step 3 Classification (w/ SVM)
- Step 3 Ensemble: Voting
  - Single window starting three seconds into the set for recognition

# Counting repetition

- Process the three-axis data to extract a one-dimensional (1d) signal
  - An elliptical band pass filter (0.15Hz – 11Hz) removes high- and low-frequency components
  - Subtract the mean from the data, apply Principal Component Analysis (PCA), and project the data onto **its first PC**
    - By projecting onto this axis – **the axis of highest variation during this exercise** – we simplify counting repetitions to the problem of counting peaks on a 1d signal

# Counting repetition

- Peak Detection



Easy: each repetition corresponds to a strong peak, all peaks are similar, and peaks occur at a constant rate.

multiple peaks per repetition

variation in the repetition rate

variation in peak shape and amplitude



# Counting repetition

- Peak Detection

- Local period estimation to reject peaks that do not correspond to actual repetitions
- Capture two key intuitions:
  - Peaks that are significantly “off-schedule” with respect to the current repetition rate are likely to be false repetitions
  - Peaks that are significantly smaller than the typical peaks in the set are likely to be false repetitions
  - Minimum and maximum reasonable times needed for one repetition (**minPeriod** & **maxPeriod**)
  - For example, it is nearly impossible for a sit-up to take less than 0.75s or more than 4s

# Results

## Segmentation Performance

	Precision	Recall
<b>End-to-End Study</b>		
Traditional	100%	100%
Close	98.8%	98.8%
Tight	85.6%	85.6%
<b>Leave-One-Out (Training Data)</b>		
Traditional	99.1%	98.3%
Close	91.4%	91.0%
Tight	86.8%	86.9%

## Accuracy (training data)

Circuit	Online	Offline	Chance
Study A	99.4%	100%	25%
Study B	98.6%	99.3%	25%
Medium	96.2%	98.1%	14.3%
Large	93.8%	96.0%	7.7%

## Accuracy (End-to-end)

Circuit	Online	Offline	Chance
A	100%	100%	25%
B	98.8%	96.3%	25%
Mean	99.3%	98.2%	25%

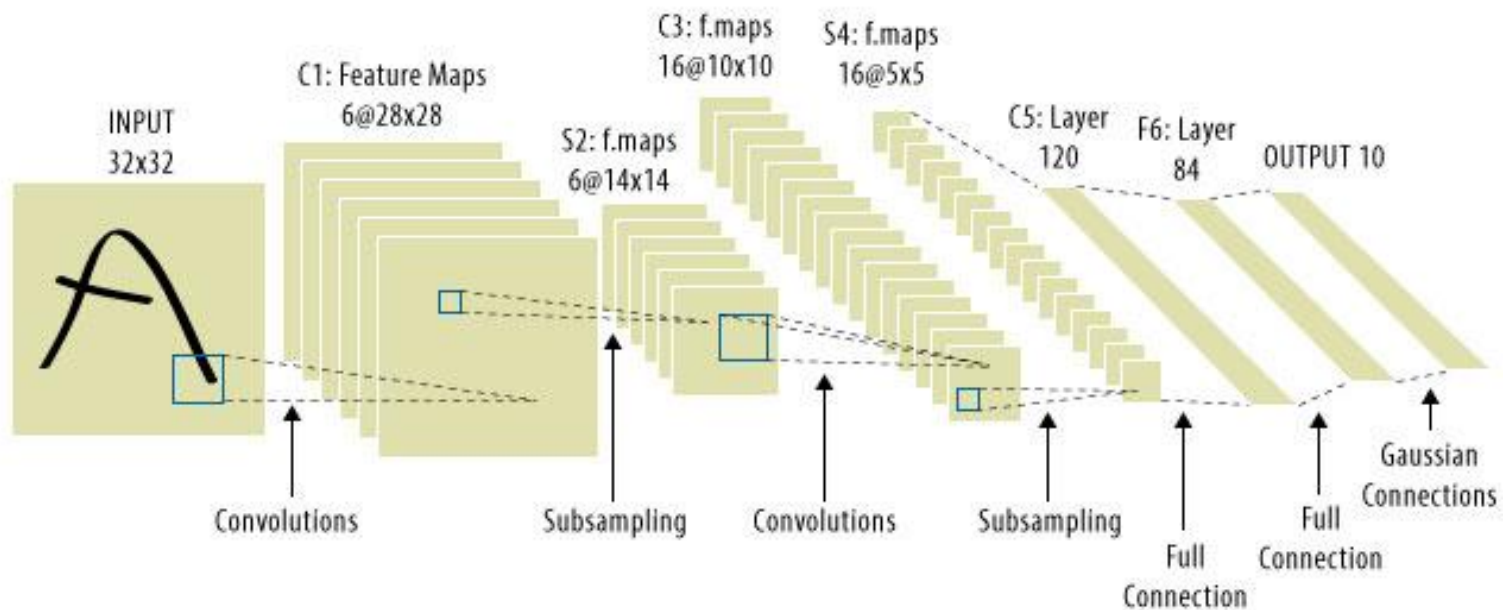
End-to-end Study: recruited additional 20 participants for data collection

# Using Deep Learning for Activity Recognition

- Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition, Sensors 2016
- CNN-based sensor fusion techniques for multimodal human activity recognition, ISWC 2017

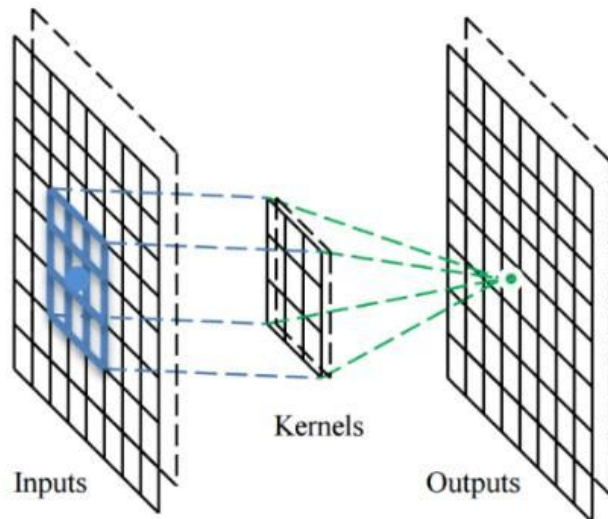
# Convolution Filter

- A CNN with a single layer extracts features from the input signal through a convolution operation of the signal with a filter (or kernel)

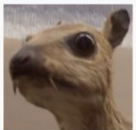



# Convolution Filter

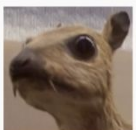
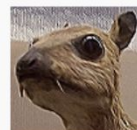
- A CNN with a single layer extracts features from the input signal through a convolution operation of the signal with a filter (or kernel)



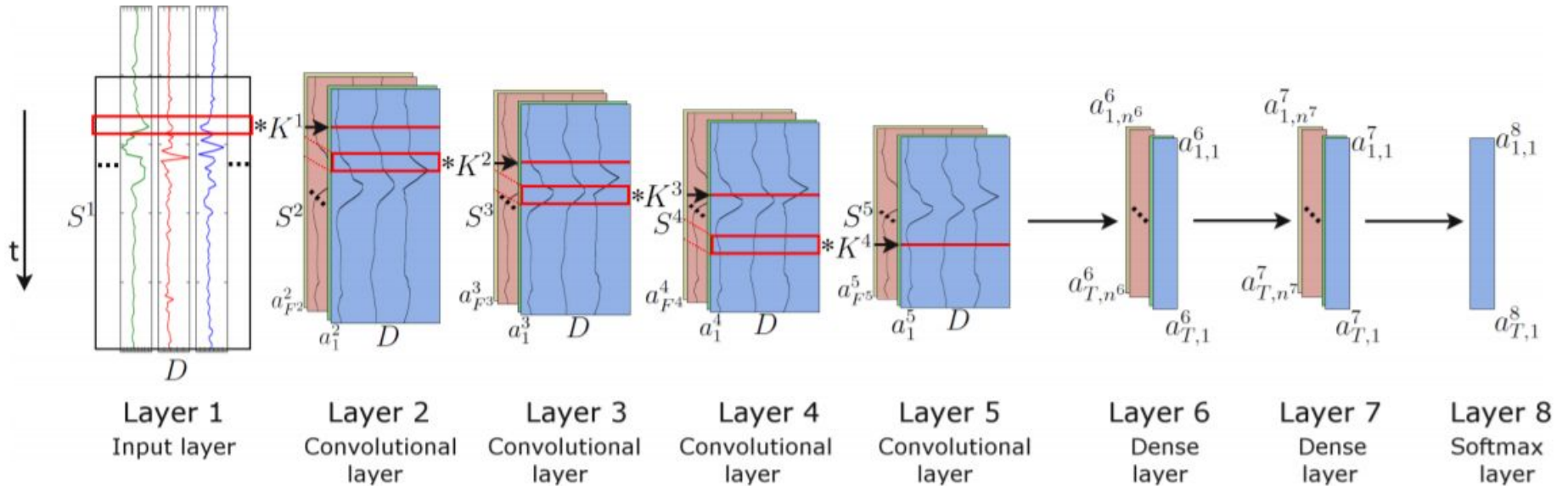
*Edge detection*

 \*  $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$  =  Kernel

*Sharpen*

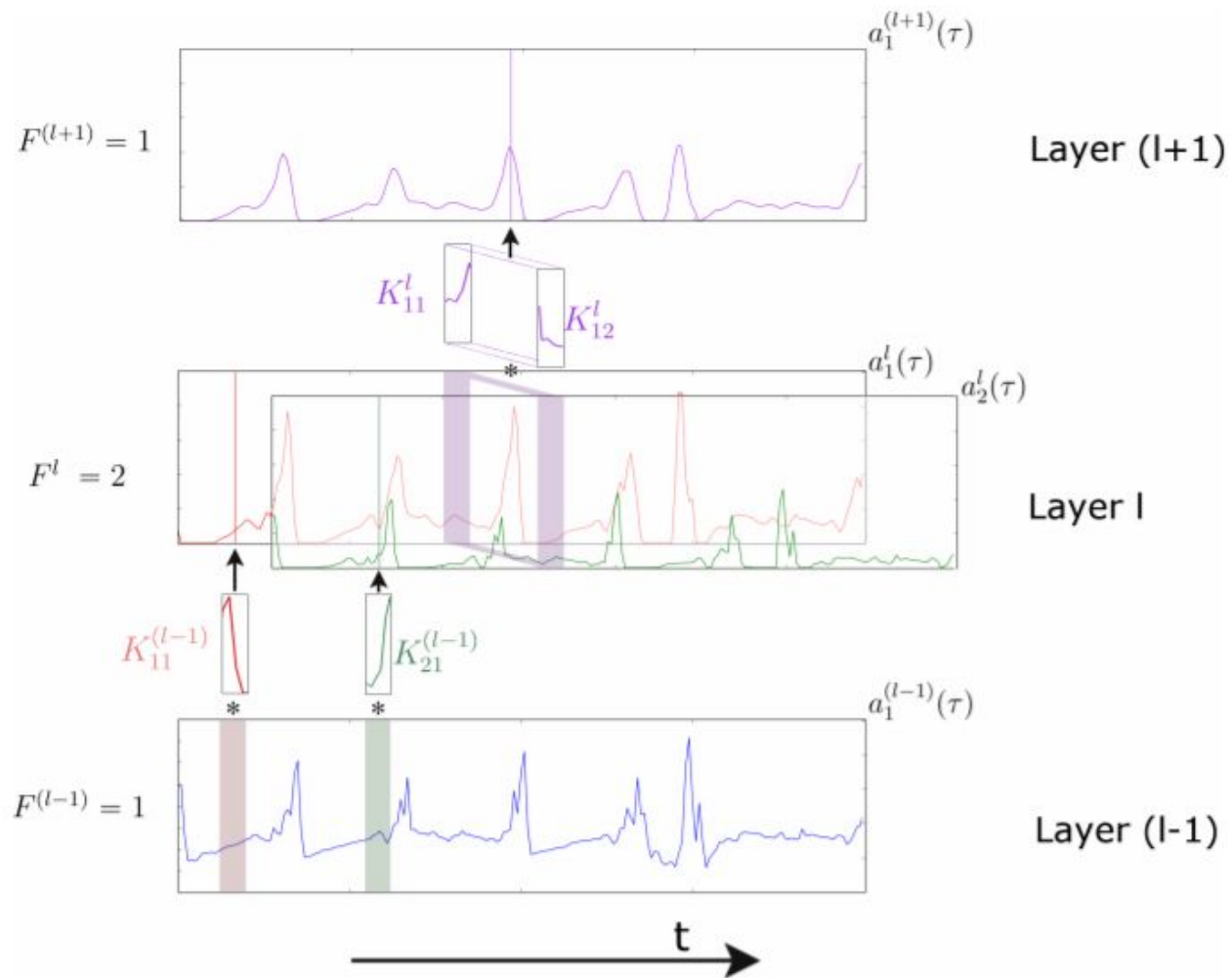
 \*  $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$  = 

# DeepConvLSTM

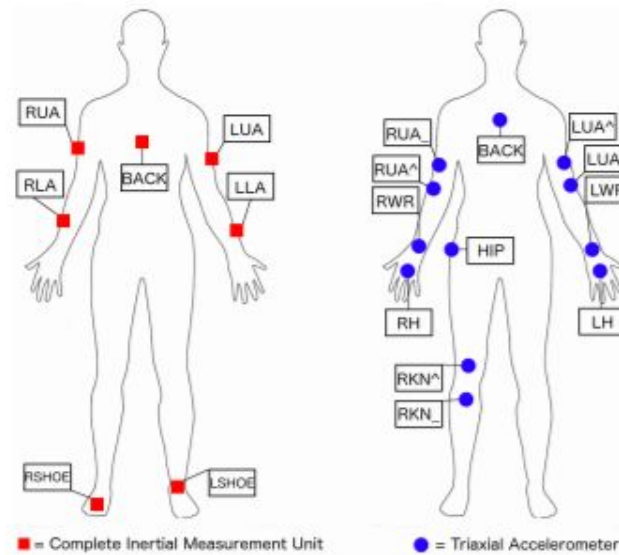


Input at Layer 1 corresponds to sensor data of size  $D \times S^1$ , where  $D$  denotes the number of sensor channels and  $S^l$  the length of features maps in layer  $l$ . Layers 2–5 are convolutional layers.  $K^l$  denotes the kernels in layer  $l$  (depicted as red squares).  $F^l$  denotes the number of feature maps in layer  $l$ . In convolutional layers,  $a_i^l$  denotes the activation that defines the feature map  $i$  in layer  $l$ . Layers 6 and 7 are dense layers. In dense layers,  $a_{t,i}^l$  denotes the activation of the unit  $i$  in hidden layer  $l$  at time  $t$ . The time axis is vertical.





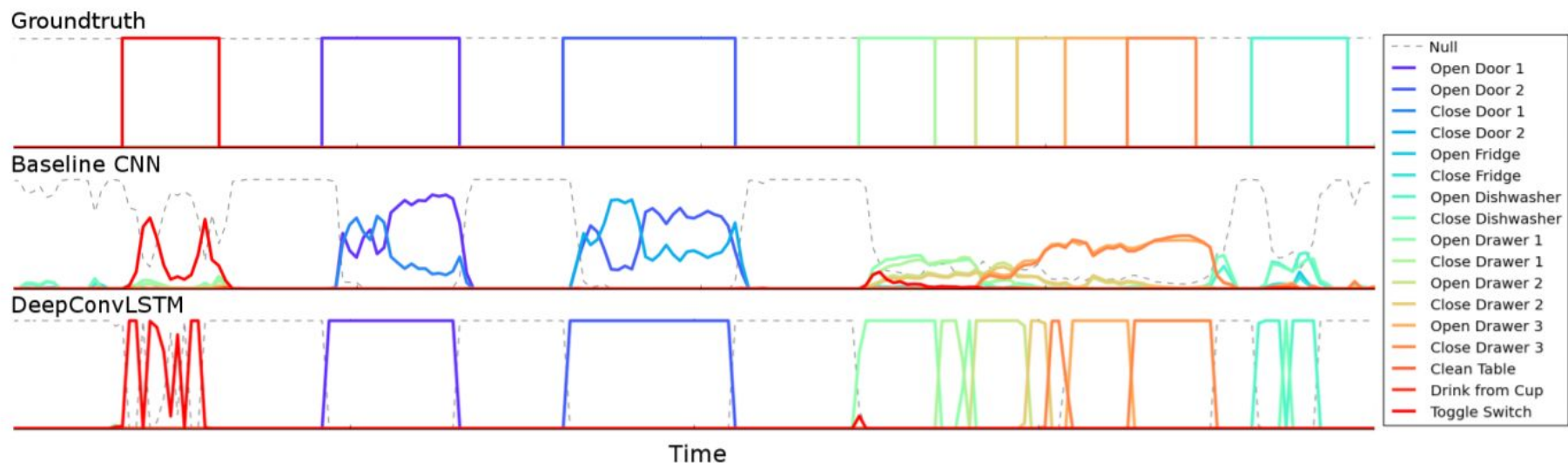
**Figure 2.** Representation of a temporal convolution over a single sensor channel in a three-layer convolutional neural network (CNN). Layer  $(l - 1)$  defines the sensor data at the input. The next layer  $(l)$  is composed of two feature maps ( $a_1^l(\tau)$  and  $a_2^l(\tau)$ ) extracted by two different kernels ( $K_{11}^{(l-1)}$  and  $K_{21}^{(l-1)}$ ). The deepest layer (layer  $(l + 1)$ ) is composed by a single feature map, resulting from temporal convolution in layer  $l$  of a two-dimensional kernel  $K_1^l$ . The time axis (which is convolved over) is horizontal.



**Figure 4.** Placement of on-body sensors used in the OPPORTUNITY dataset (left: inertial measurements units; right: 3-axis accelerometers) [7].

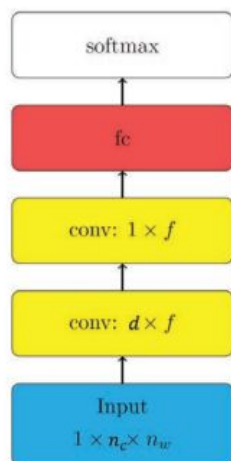
OPPORTUNITY						Skoda		
Gestures			Modes of Locomotion					
Name	# of Repetitions	# of Instances	Name	# of Repetitions	# of Instances	Name	# of Repetitions	# of Instances
Open Door 1	94	1583	Stand	1267	38,429	Write on Notepad	58	20,874
Open Door 2	92	1685	Walk	1291	22,522	Open Hood	68	24,444
Close Door 1	89	1497	Sit	124	16,162	Close Hood	66	23,530
Close Door 2	90	1588	Lie	30	2866	Check Gaps Door	67	16,961
Open Fridge	157	196	Null	283	16,688	Open Door	69	10,410
Close Fridge	159	1728				Check Steering Wheel	69	12,994
Open Dishwasher	102	1314				Open and Close Trunk	63	23,061
Close Dishwasher	99	1214				Close both Doors	69	18,039
Open Drawer 1	96	897				Close Door	70	9783
Close Drawer 1	95	781				Check Trunk	64	19,757
Open Drawer 2	91	861						
Close Drawer 2	90	754						
Open Drawer 3	102	1082						
Close Drawer 3	103	1070						
Clean Table	79	1717						
Drink from Cup	213	6115						
Toggle Switch	156	1257						
Null	1605	69,558						





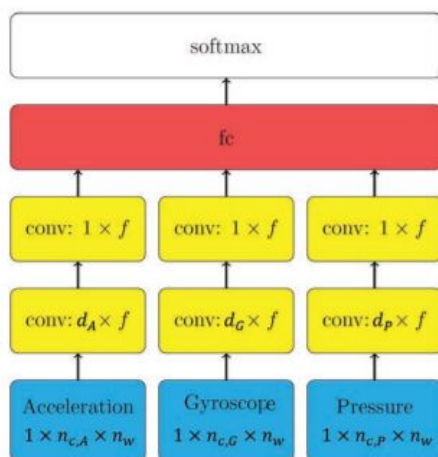
**Figure 6.** Output class probabilities for a ~25 s-long fragment of sensor signals in the test set of the OPPORTUNITY dataset, which comprises 10 annotated gestures. Each point in the plot represents the class probabilities obtained from processing the data within a sequence of 500 ms obtained from a sliding window ending at that point. The dashed line represents the *Null* class. DeepConvLSTM offers a better performance identifying the start and ending of gestures.

# Sensor Fusion w/ Deep Learning



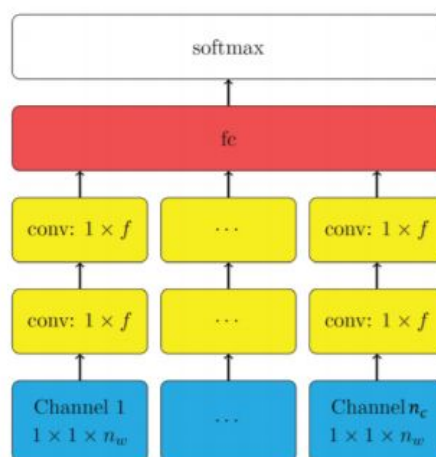
(a) Early fusion

Early fusion (EF)



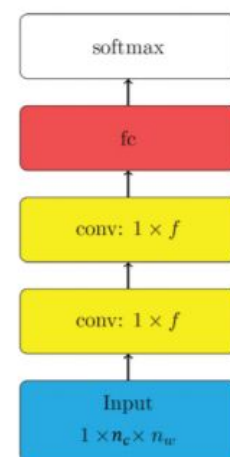
(b) Sensor-based late fusion

Sensor-based late fusion (SB-LF)



(c) Channel-based late fusion

Channel-based late fusion (CB-LF)



(d) Shared filters hybrid fusion

Shared filters hybrid fusion (SF-HF)

*the same filters are used for all input channels*

Fusion technique		EF	SB-LF	CB-LF	SF-HF
2L-CNN	mean	0.74	0.76	0.81	<b>0.86</b>
	std	0.013	0.043	0.045	0.034
3L-CNN	mean	0.74	0.81	0.81	<b>0.85</b>
	std	0.016	0.052	0.05	0.029

**Table 3. Average  $F_1$ -scores achieved on PAMAP2 for different sensor fusion models (EF, SB-LF, CB-LF and SF-HF) with two and three layered CNNs using zNorm+BN normalization.**

# DL Architecture

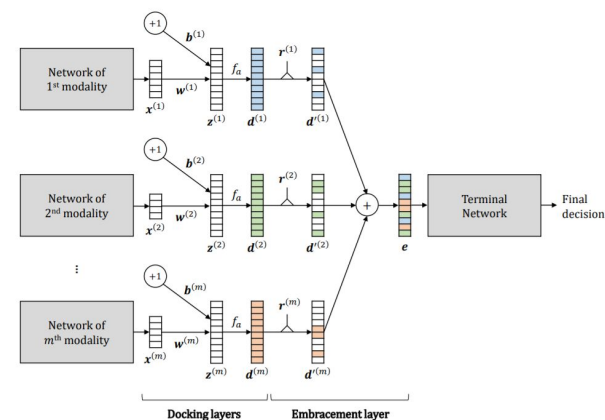
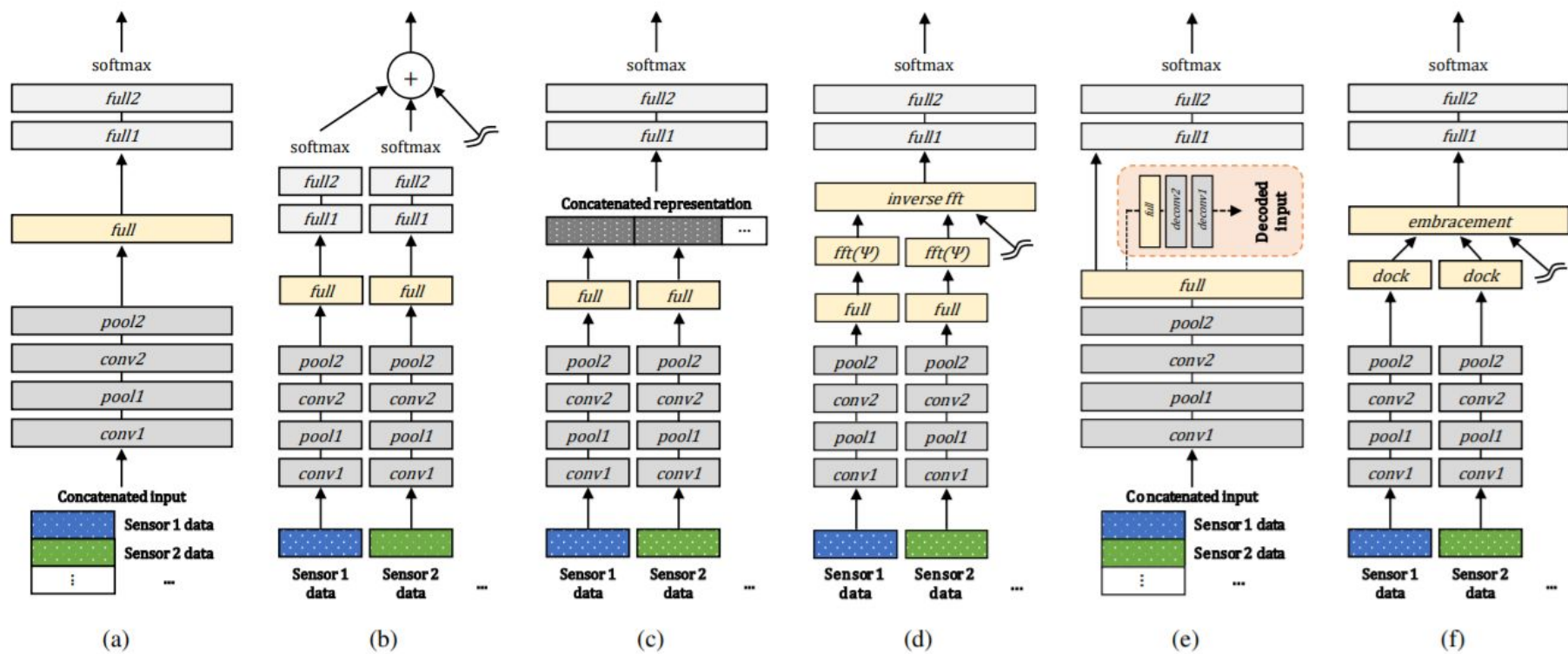


Figure 3: Structures of the employed network models for the gas sensor arrays dataset. (a) Early integration (b) Late integration (c) Intermediate integration (d) Compact multi-linear pooling [50] (e) Multimodal autoencoder [41] (f) EmbraceNet

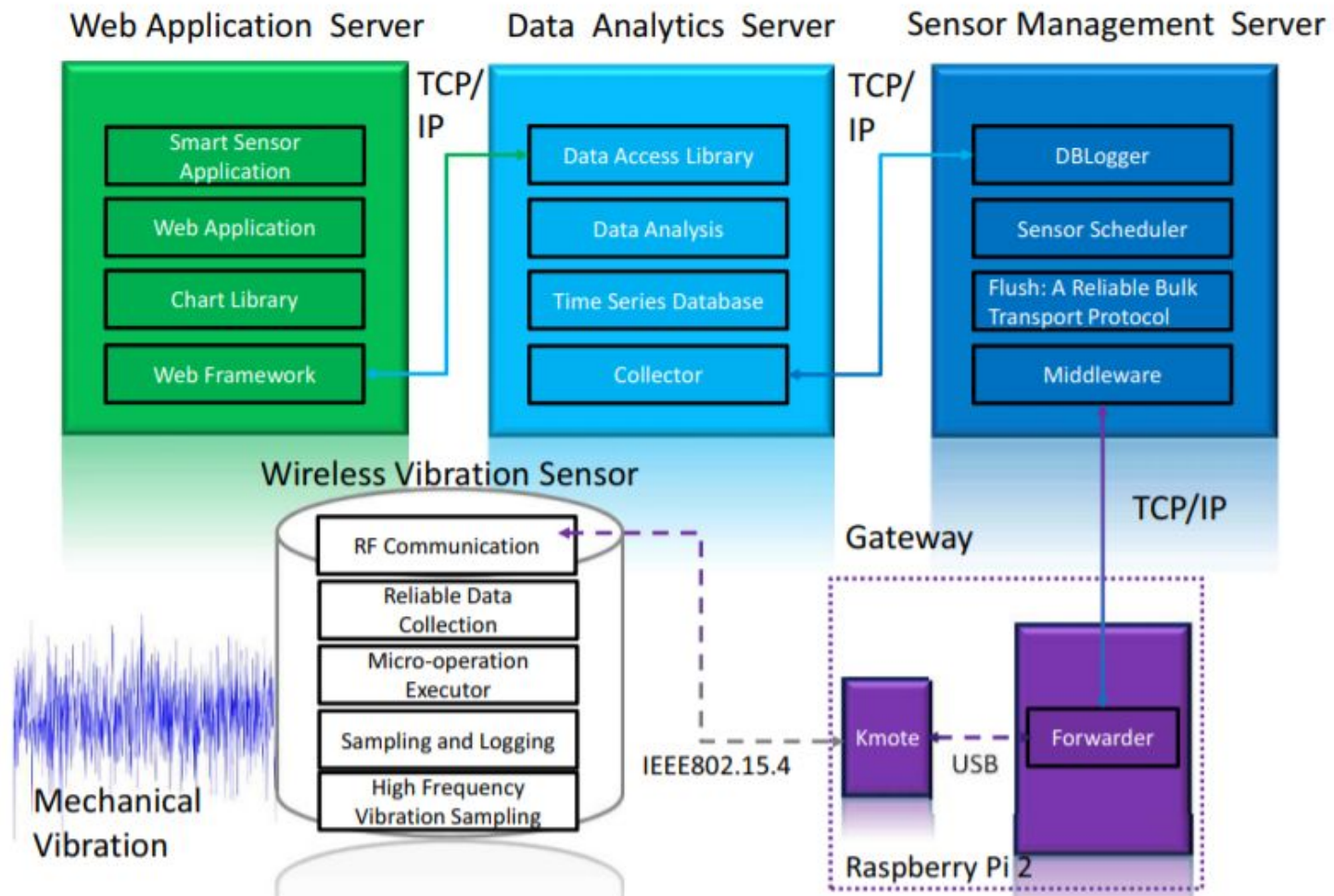
# **Vibration Analysis for IoT Enabled Predictive Maintenance**

Deokwoo Jung, Zhenjie Zhang, Marianne Winslett

ICDE 2017



# Remaining Usefulness Lifetime (RUL) estimation



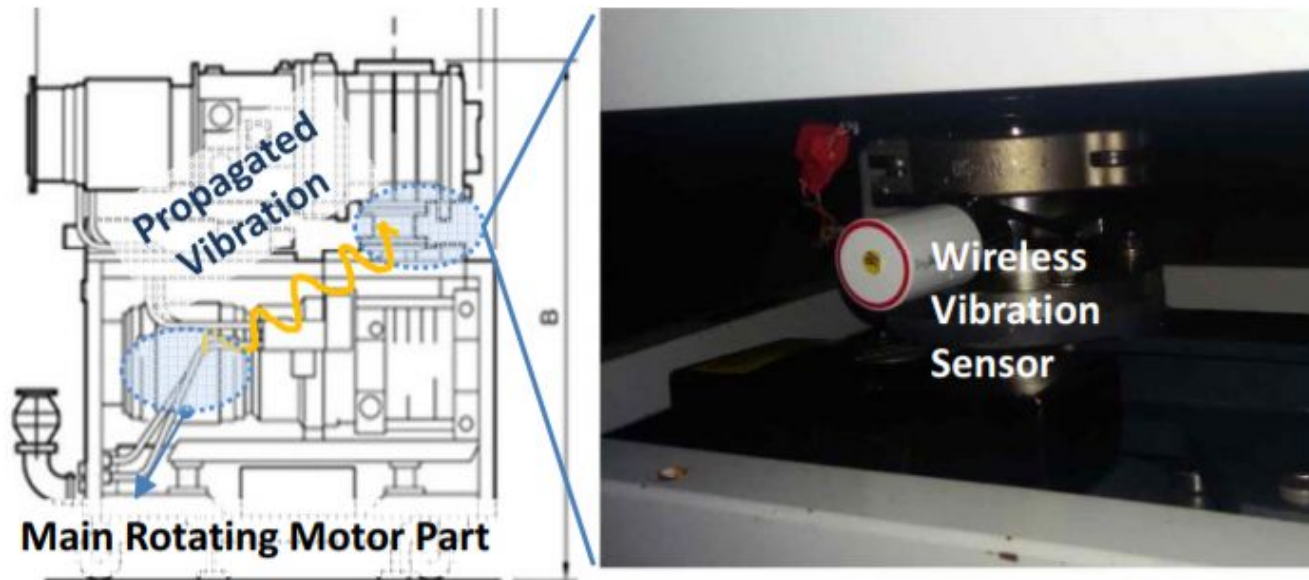
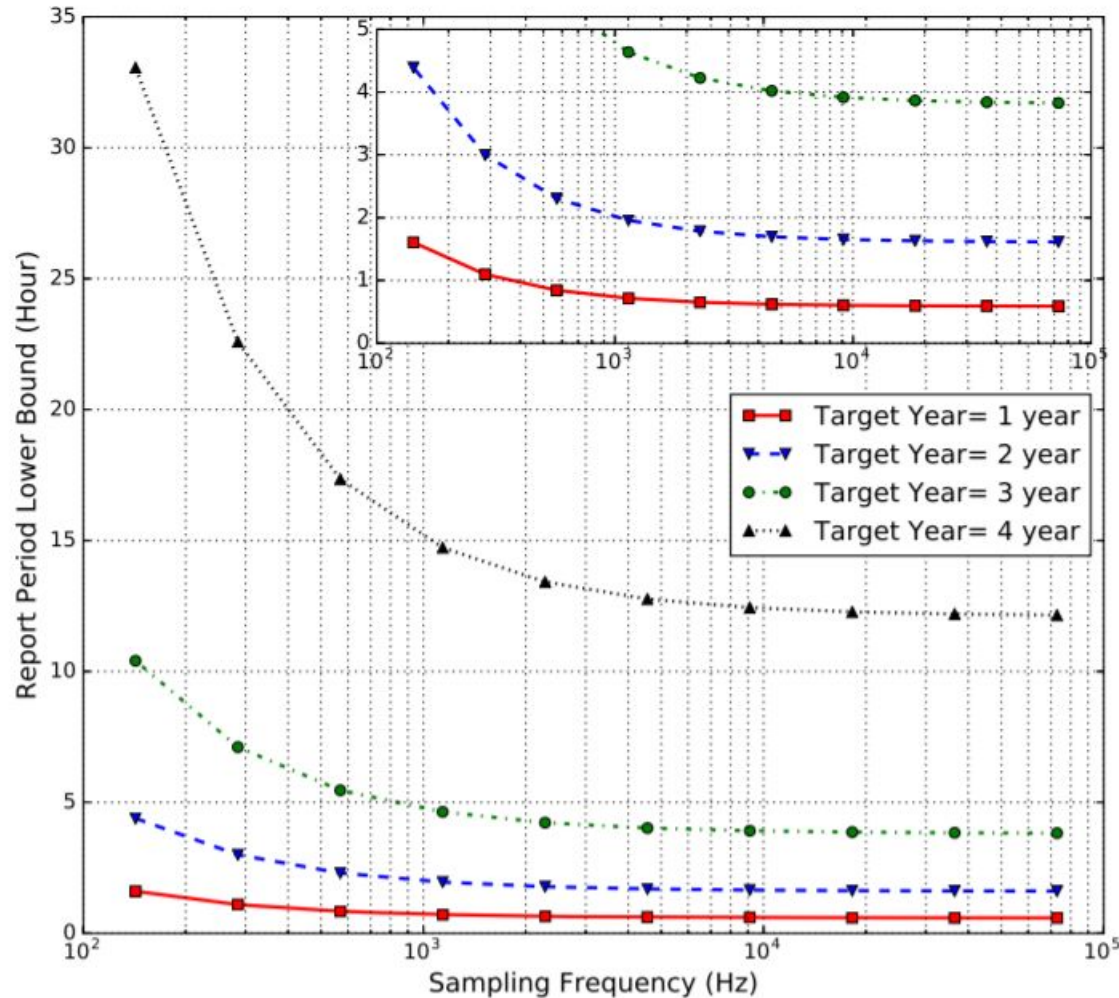


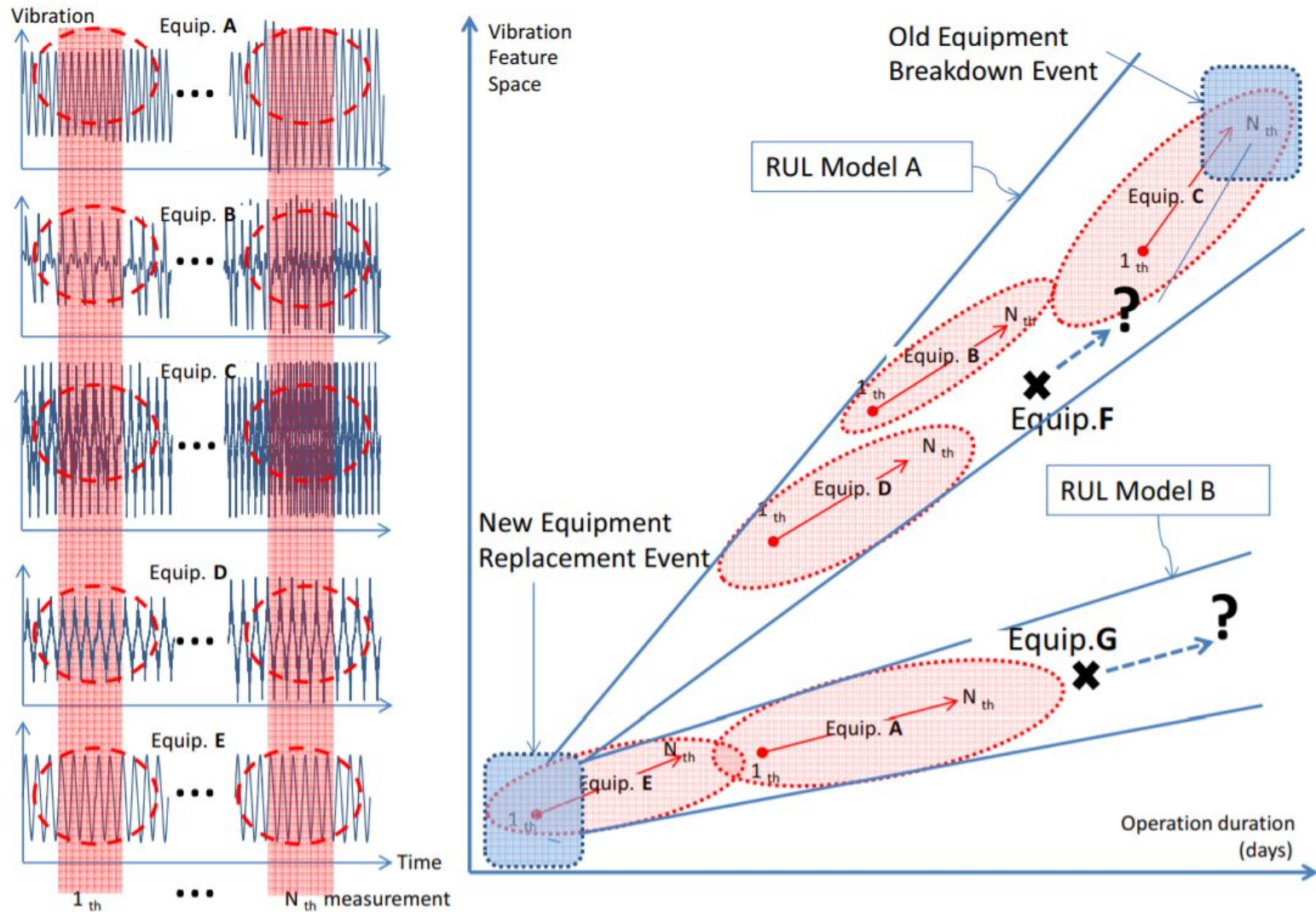
Fig. 2: Non-Intrusive Vibration Monitoring on Pump's Suction Connector using Wireless Vibration Sensor

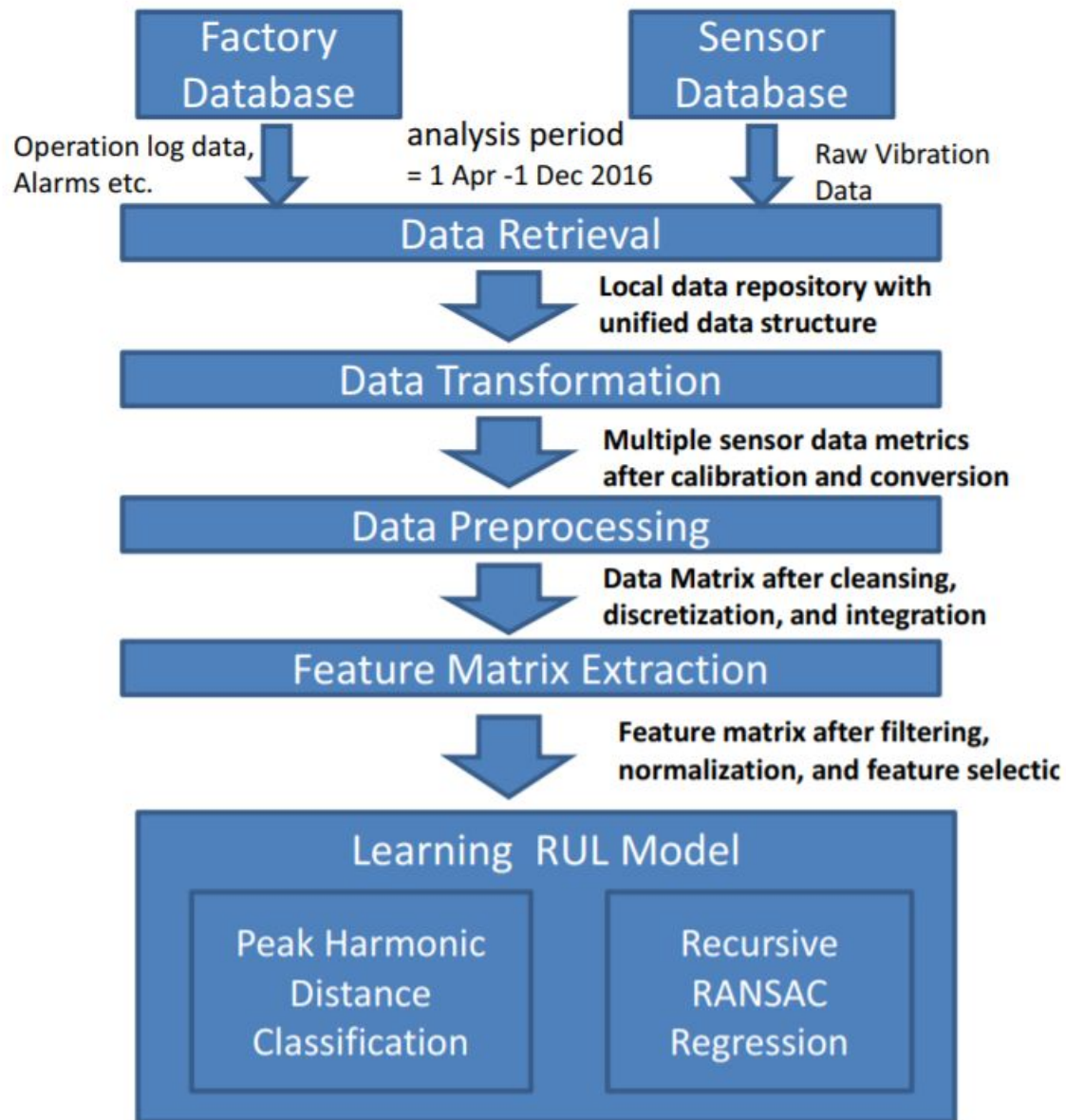
# Wireless Sensing: Trade-offs





# Remaining Usefulness Lifetime





# Features

- Root mean square (or RMS)
  - RMS is commonly used to measure the overall magnitude of vibrations
- Power spectral density (or PSD)
  - PSD allows us to better capture the subtle characteristics of the vibration behavior, by identifying detailed harmonic components behind RMS

# Labels: Machine Condition Zones

- **C1: Zone A**

- Vibration of newly Commissioned machines

- **C2: Zone B**

- Acceptable for unrestricted long -term operation

- **C3: Zone C**

- Unsatisfactory for long term continuous operation

- **C4: Zone D**

- Vibration of sufficient severity to cause damage to machine

# Estimating Deviation

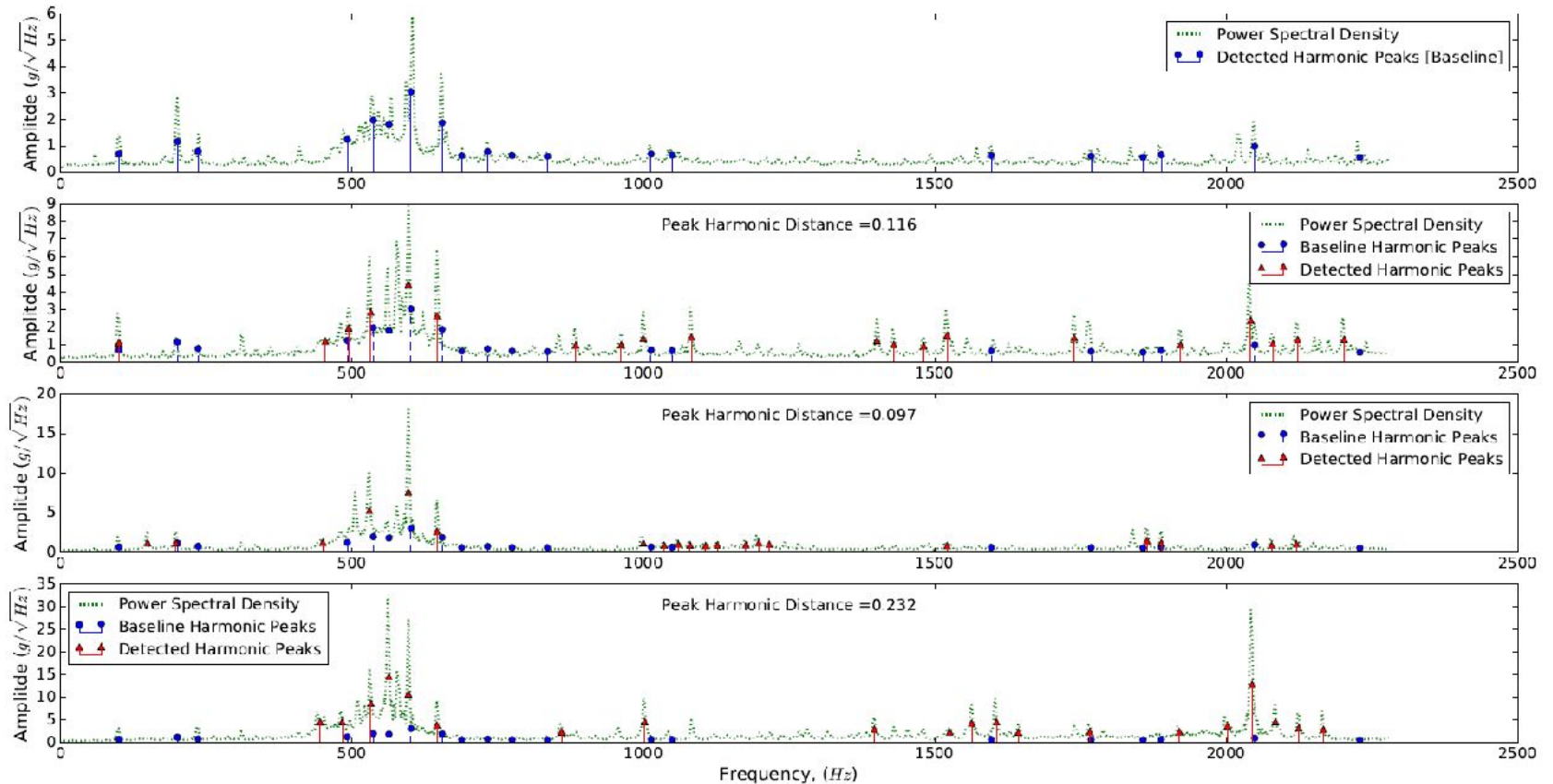


Fig. 9: Peak harmonic feature distance comparison: (Top) a PSD sample and its peak harmonic feature taken from a healthy condition, Zone A [baseline peak harmonic feature] (from the Second to the Bottom) Comparison of peak harmonic distances from the baseline feature (on Top) for PSD samples randomly chosen taken other Zones.

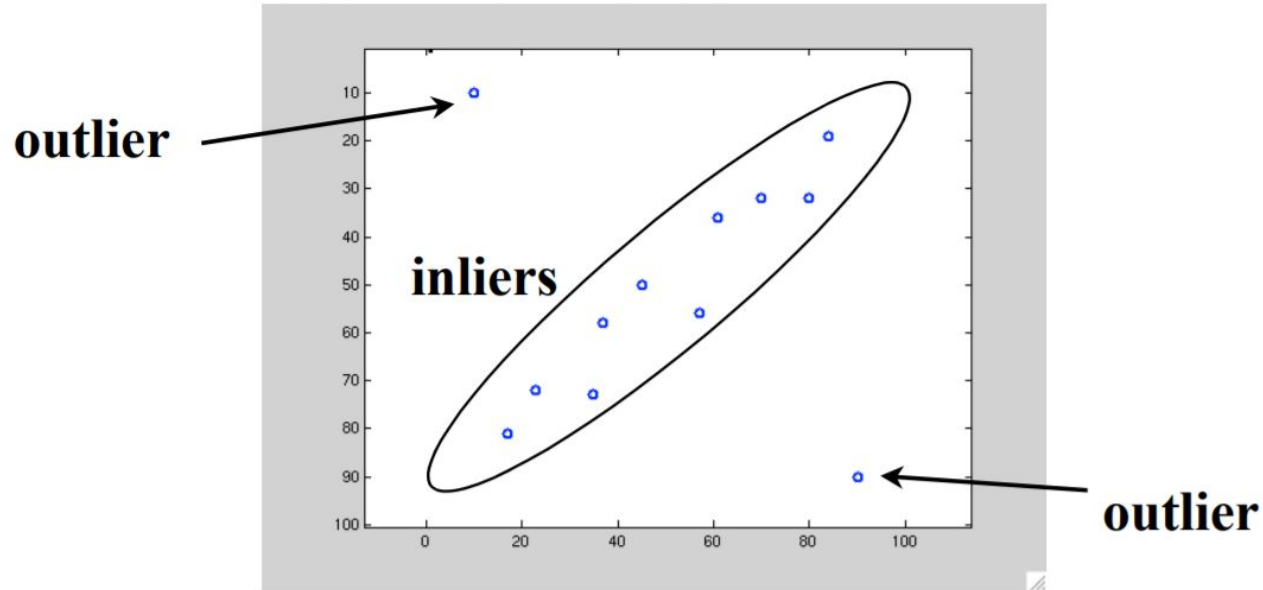
# Zone Classification

- Key idea: classifying measurements from individual equipment's into their corresponding health condition category given their harmonic distance from an exemplary measurement sample from healthy condition, Zone A
- Let  $D_a$  denote the Peak Harmonic Distance from Zone A
  - $D_a$  is a measure to quantify how dissimilar the harmonic peaks from equipment from normal harmonic peaks trained for zone A
- The Peak Harmonic Distance Classification algorithm constructs a classifier using training data to determine equipment's health condition category given the observed measurement's  $D_a$

# RUL Estimation

- $D_a$  is expected to monotonically increase over equipment's service time unless maintenance actions are taken
- Many maintenance actions and other operational events add outliers into the monotonic increasing model of  $D_a$
- Use Random sample consensus (RANSAC) approach
  - The Recursive RANSAC regression algorithm iteratively runs the RANSAC algorithm on outliers until no more monotonically increasing linear model could be found any more
- RUL prediction
  - First learn the threshold  $D_a$  between zone C and zone D
  - This threshold is chosen to minimize the error of wrongly classifying records in zone C and zone D
  - The algorithm then determines the equipment's RUL by projecting the equipment's features over the space and checking if the projection exceeds the threshold

**Loosely speaking, outliers are points that don't "fit" the model.  
Points that do fit are called "inliers"**



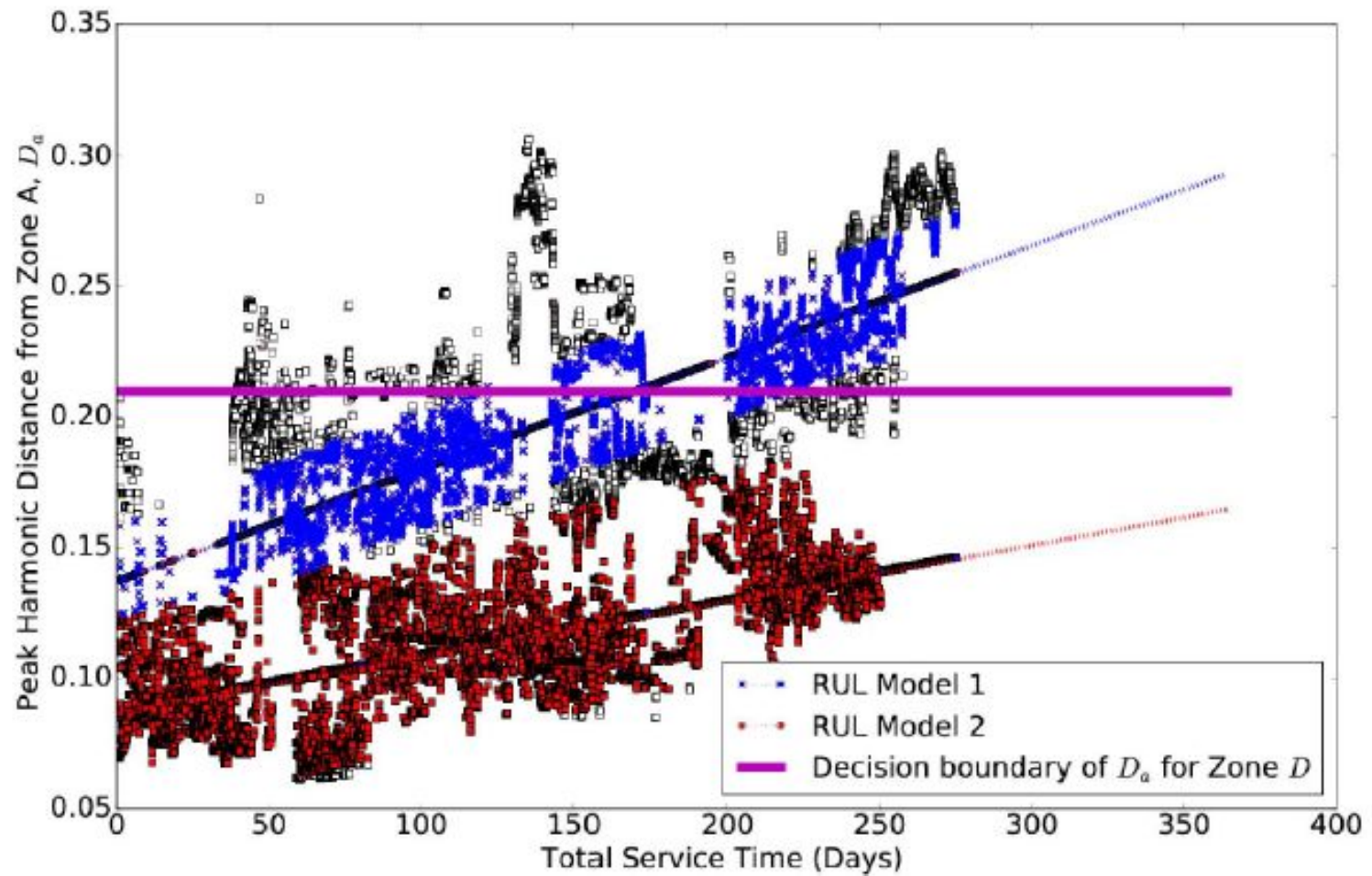
- Robust estimation
  - **Phase I:** Classify data points as outliers or inliers
  - **Phase II:** Fit a linear model to inliers while ignoring outliers
- Example: RANSAC (RANDOM Sample Consensus)



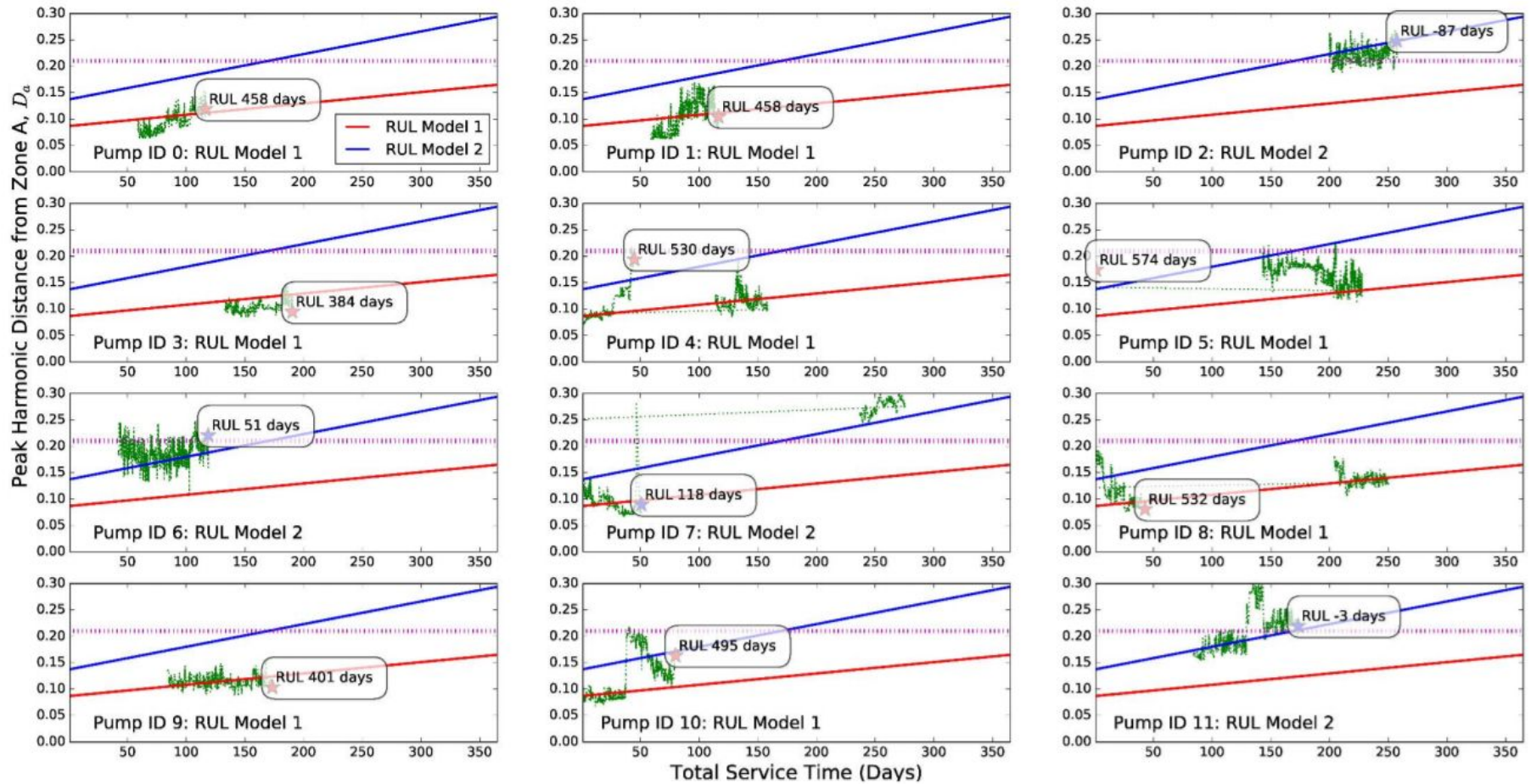
# RUL Estimation

- $D_a$  is expected to monotonically increase over equipment's service time unless maintenance actions are taken
- Many maintenance actions and other operational events add outliers into the monotonic increasing model of  $D_a$
- Use Random sample consensus (RANSAC) approach
  - The Recursive RANSAC regression algorithm iteratively runs the RANSAC algorithm on outliers until no more monotonically increasing linear model could be found any more
- RUL prediction
  - First learn the threshold  $D_a$  between zone C and zone D
  - This threshold is chosen to minimize the error of wrongly classifying records in zone C and zone D
  - The algorithm then determines the equipment's RUL by projecting the equipment's features over the space and checking if the projection exceeds the threshold

# Results



# Results



# **Using Mobile Phones to Determine Transportation Modes**

Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin,  
Mark Hansen, and Mani Srivastava

ACM Transactions on Sensor Networks  
Vol. 6, No. 2, Article 13, Feb 2010

# Transportation Modes

- Using mobile phones to determine a user's transportation mode when outside (e.g., stationary, walking, running, biking, or in motorized transport)



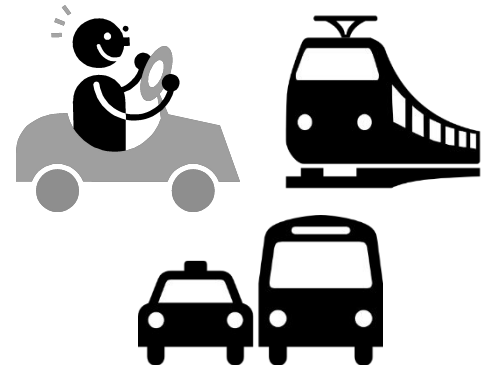
Walking



Running

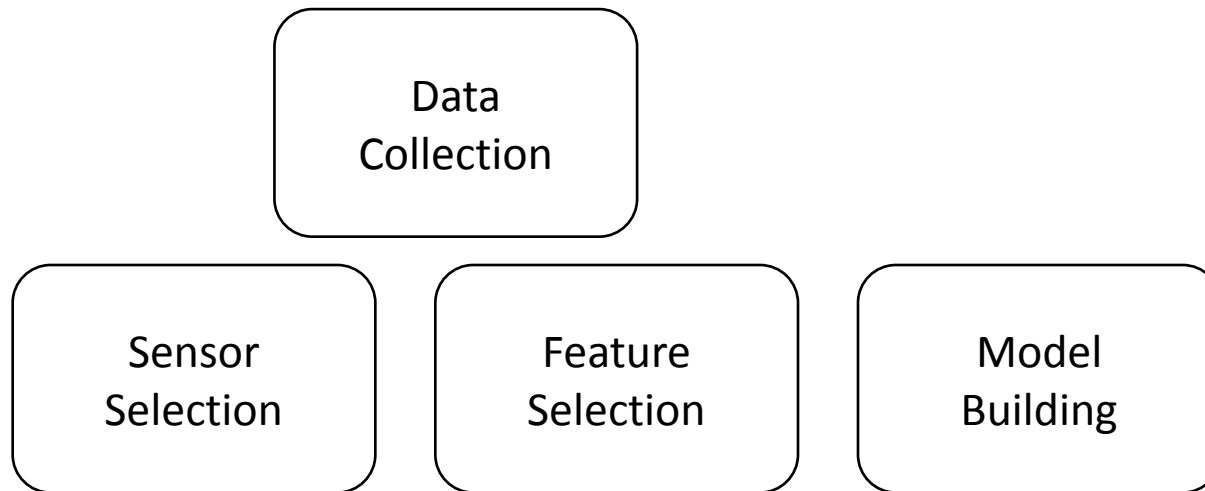


Biking



Motorized Transport

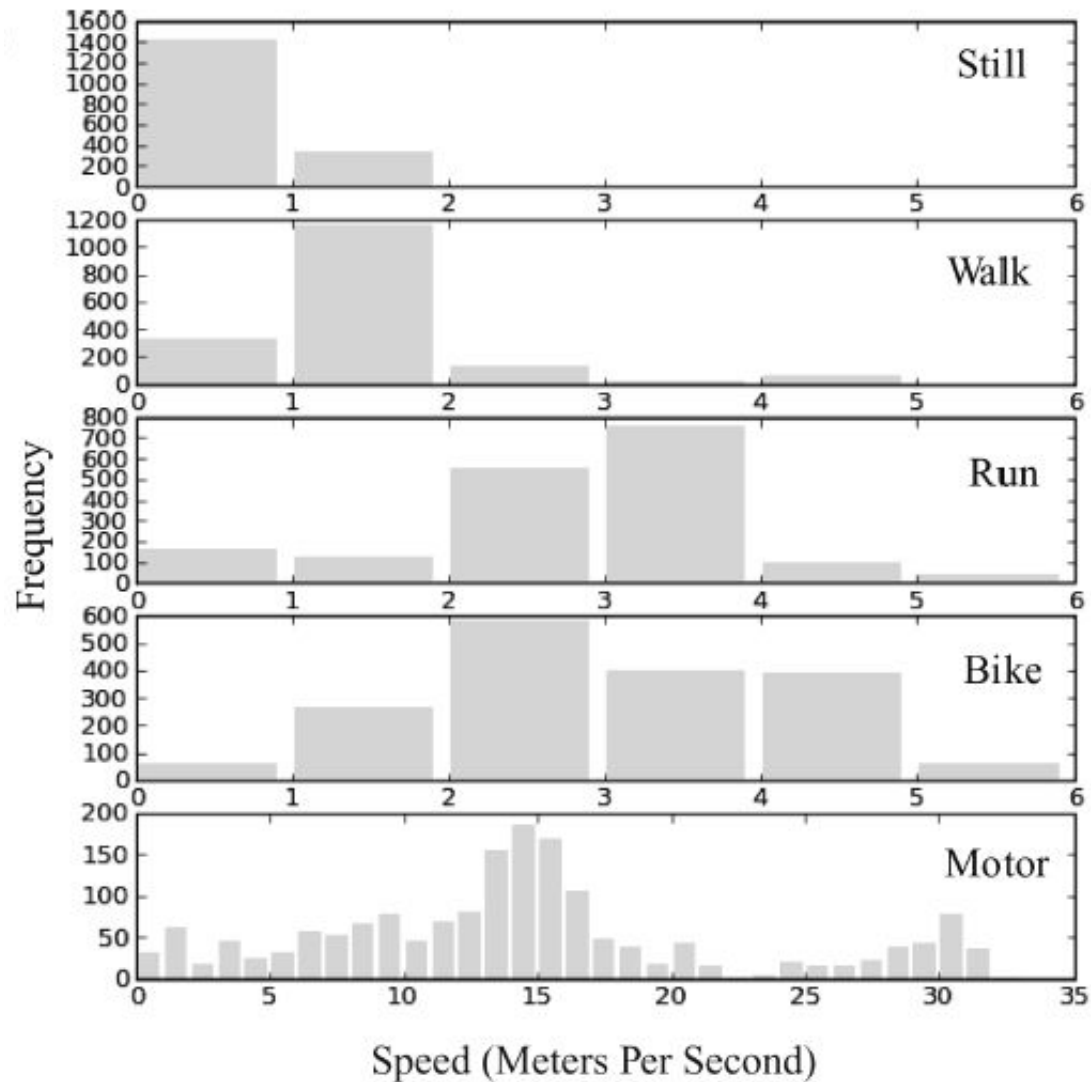
# Procedure



# Sensors?

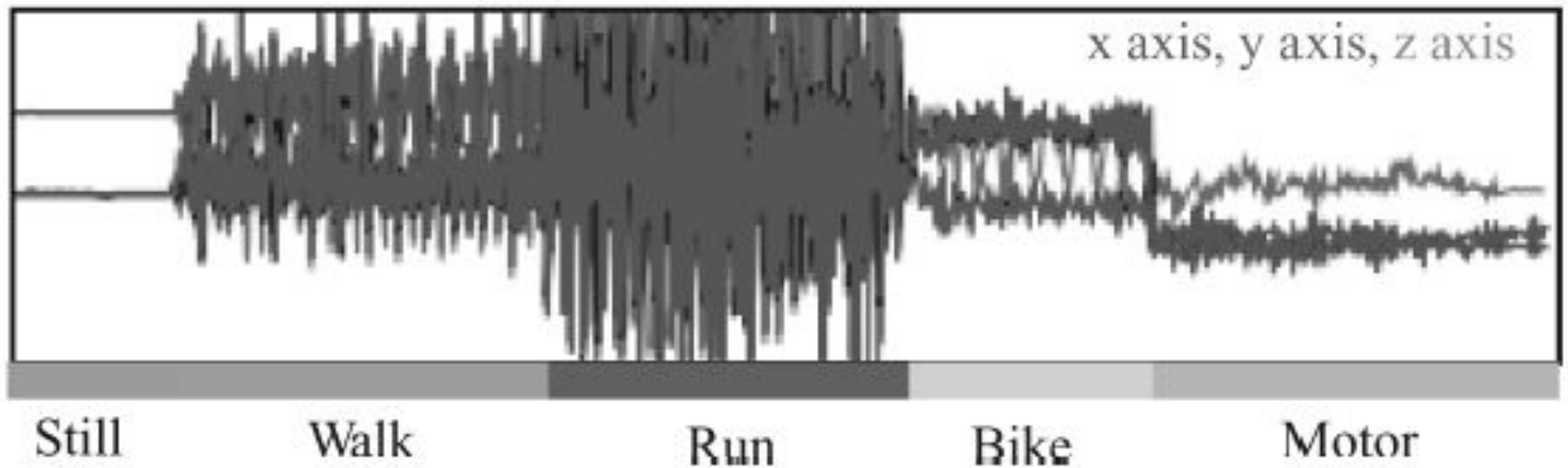
- Motion sensors
  - **Accelerometer**
  - Gyroscope
  - Magnetometer
- Environment sensor
  - Barometer
  - Light
  - Temperature/humidity
- Location sensors
  - **GPS – location, speed, altitude**
  - Wireless signatures (e.g., Bluetooth, WiFi, cell-tower info, etc.)

# Sensors – speed & motion





# Sensors – speed & motion



# Features

- Window size: 1 second
  - Too small window may not be effective if sampling frequency is low
  - Too large window suffers from noise as it may span multiple activities
- Features:
  - Mobility: accelerometer
    - Magnitude of a force vector => mean, variance, energy
    - DFT (discrete Fourier transform): energy coefficients between 1-10 HZ (appropriate for detecting pedestrian based motion)
  - Speed: GPS
    - Speed from GPS samples
    - Removing invalid points by applying noise filtering via analyzing accuracy , dilution of precision, change of speed, etc.

# Feature Selection

- Correlation-based feature selection (CFS)
  - Select a subset of features that maximize correlation with the class, but minimize correlation among features
  - Must specify search methods (e.g., best first, forward, etc.)
- Ranking each feature via information gain metric:

Feature	Score
GPS Speed	1.431
Accelerometer Variance	1.426
Accelerometer DFT (3 Hz)	1.205
Accelerometer DFT (2 Hz)	1.125
Accelerometer DFT (1 Hz)	0.915

# Classifier

- Instance classifiers:
  - Decision tree
  - K-means clustering
  - Naïve Bayes (NB)
  - Nearest neighbor (NN)
  - Support vector machine (SVM)
- HMM-based classifier:
  - Continuous HMM:
    - Output states – accelerometer / GPS features modeled as independent multi-variate Gaussian distributions
    - Hidden states – actual transportation modes
    - State transition probabilities
  - Instance based classifier + DHMM(Discrete Hidden Markov Model)
    - Output states – initial classification results
    - Hidden states – actual transportation modes
    - State transition probabilities

# Experiments

- Hardware: Nokia N95
  - CPU : 332 MHz ARM processor
  - RAM : 128MB
  - 3 axis accelerometer that can sample at 32 Hz
  - Built-in GPS receiver that can sample at 1 Hz
  - WiFi radio that can scan at 0.33 Hz
  - GSM cell radio that can sample at 1 Hz
  - Bluetooth radio that can scan at 0.08 Hz
  - Battery : 950 mAh
  - OS : Symbian S60 3<sup>rd</sup> Edition
- Software Setup
  - Weka Machine Learning Toolkit
  - Generalized Hidden Markov Model library
  - Python scripting



# Experiments

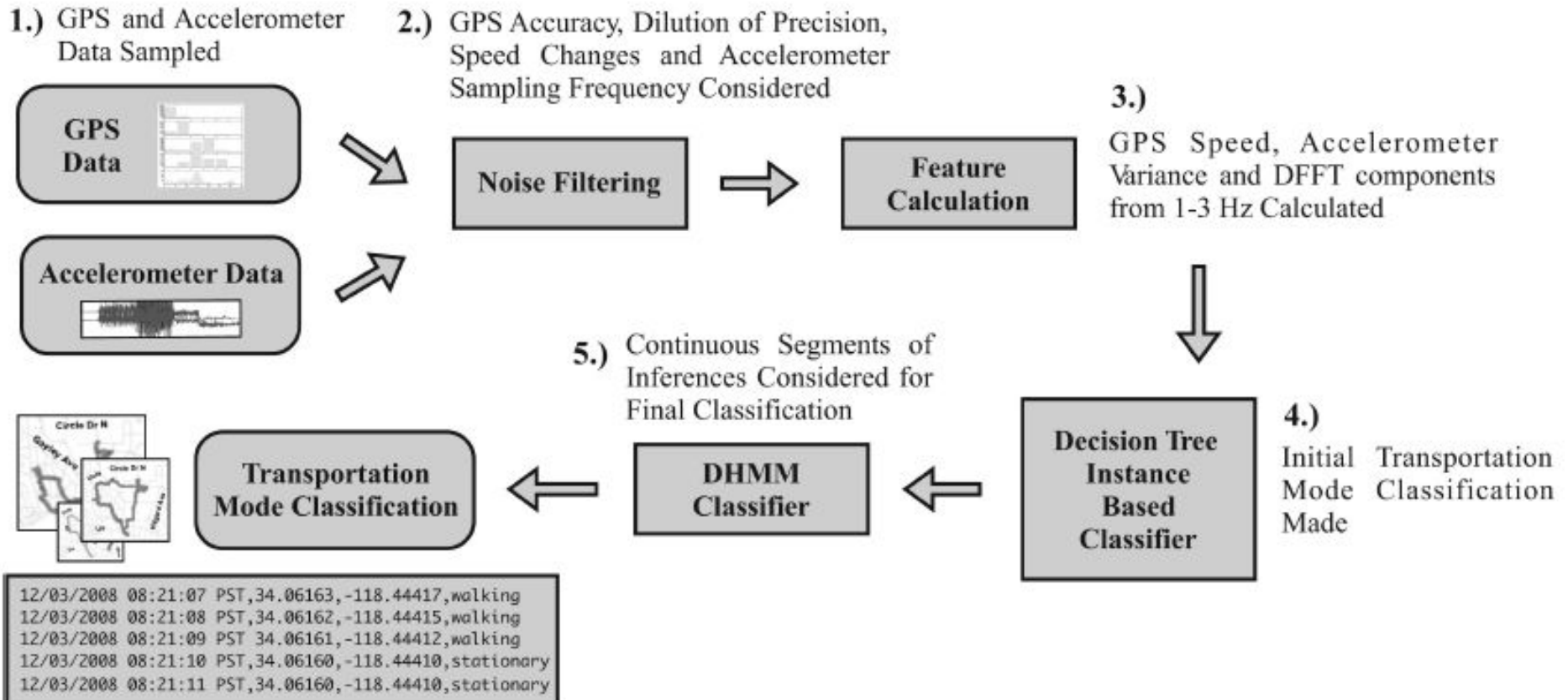
- Data Collection
  - 16 individuals (8 male + 8 female); age: 20-45
  - Attached six phones: arm, wrist, waist, chest, hand, pocket, bag
  - Collected data: accelerometer, GPS, WiFi, GSM info
  - Each participant performed 5 activities one by one
  - Each participant's data collection duration: about 1.25 hours

# Experiments - Accuracy

Table IV. Precision Results for Classifiers

	Still	Walk	Run	Bike	Motor	All
<b>DT</b>	<b>95.0</b>	<b>87.6</b>	<b>95.5</b>	<b>84.5</b>	<b>93.9</b>	<b>91.3</b>
KMC	54.0	81.0	98.5	45.6	98.9	75.6
NB	88.4	88.1	93.5	75.6	71.3	83.4
NN	96.4	87.3	93.3	84.8	92.7	90.9
SVM	90.7	88.8	95.9	81.6	97.8	91.0
CHMM	89.2	90.0	94.3	80.5	77.6	86.3
<b>DT-DHMM</b>	<b>95.5</b>	<b>92.4</b>	<b>96.4</b>	<b>87.9</b>	<b>96.2</b>	<b>93.7</b>

# Structure of Overall Classifier





# Experiments – Impacts of device placement variation

- Mobile phones are often carried at different positions
- Classifier is trained on data from all 6 positions
  - Arm, bag, chest, hand, pocket, and waist

	DT+HMM
All Positions	93.6
Arm	94.9
Bag	94.8
Chest	94.5
Hand	95.0
Pocket	94.3
Waist	94.4

# Experiments – Resource overhead

- Measurement using Nokia's profiler

Memory and CPU Benchmarks

Activity	CPU %	RAM (MB)
Phone Idle	2.18	28.91
Active Call	2.31	30.00
Music Player	30.86	30.26
Video Player	14.63	32.58
Game Playing	97.34	37.52
<b>Transport Mode</b>	<b>6.91</b>	<b>29.64</b>

Energy consumption

Activity	Power (Watts)
Phone Idle	0.054
Cell Sampling	0.056
Accelerometer Sampling	0.111
Bluetooth Sampling	0.233
GPS Assisted Lock	0.718
GPS Normal Lock	0.407
GPS Sampling	0.380
WiFi Sampling	0.230
Music Player	0.447
Video Player	0.747
Active Call	0.603
Gaming	1.173
<b>Transport Mode</b>	<b>0.425</b>

# Experiments – Personalized models

- Personalized as a model is trained only using each user's dataset
- Slight improvement (2-3%) as opposed to multi-user based model

User 1	94.5	User 9	96.0
User 2	97.9	User 10	97.6
User 3	93.3	User 11	93.4
User 4	93.8	User 12	96.6
User 5	92.7	User 13	96.3
User 6	96.8	User 14	96.8
User 7	98.0	User 15	95.2
User 8	97.1	User 16	96.7
		Average	95.8

(DT-DHMM model)

# Summary

- Transportation mode inference: stationary, walking, running, biking and in motorized travel
- Using a mobile phone equipped with a GPS receiver and an accelerometer
- Best performing model: Decision Tree + DHMM
  - Achieving a high accuracy level (93.6%)
  - Less influenced by phone orientation, user variation