Hit spacebar to record Sample 3/5 of gesture 'Backhand Tennis'

PROTOTYPING WITH ML 2: SHAPE MATCHING

CSE 599 Prototyping Interactive Systems | Lecture 12 | Nov 5

Jon Froehlich • Liang He (TA)

DESIGN USE BUILD

MAKEABILITY LAB

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY of WASHINGTON

A2 SHARE OUTS & CRITIQUE

A2: Fabrication: 3D-Printed Interactive Night Light

Published | Edit

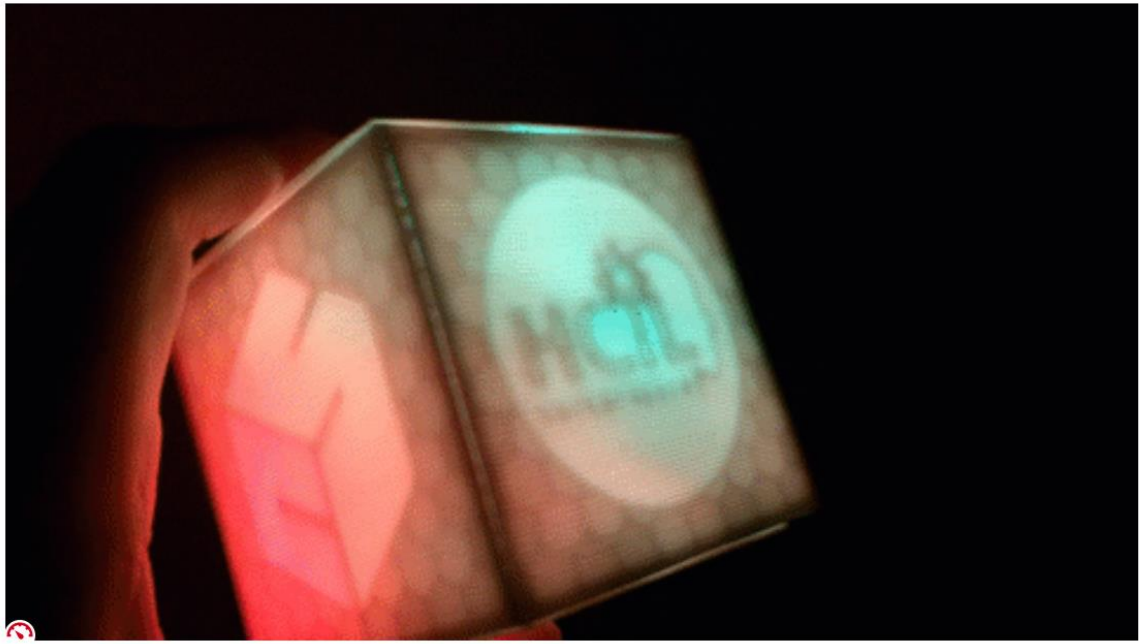


Image caption: The *Tangible Interactive Computing* Top Maker Award from CMSC838f, Spring 2015 designed by Jon Froehlich based on the Holocron Nightlight by CMSC838f student Philip Dasler.

Overview

In this assignment, you will design and fabricate a 3D-printed interactive night light, which responds to user interaction, creatively diffuses the light (*e.g.*, playfully or elegantly), and fully encloses your Arduino and electronics. The specific model is up to you but should include: (1) a mounting stand for the internal electronics; (2) a carefully measured and tightly fit input slot for the USB micro cable to power your design; (3) and similarly well-designed fittings for any input controls you want to expose. You will likely need to design and print a multi-part model, which

Related Items

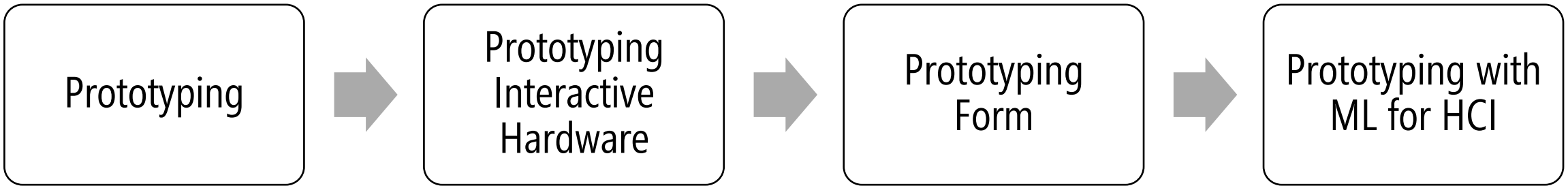
SpeedGrader™

Download Submissions

0 out of 9 Submissions Graded

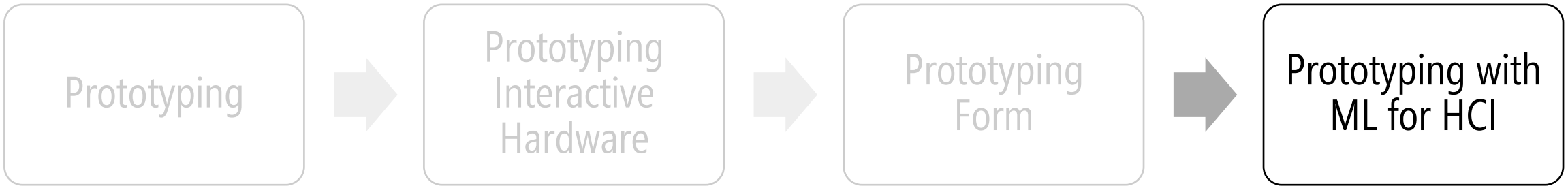
PROTOTYPING INTERACTIVE SYSTEMS

COURSE CURRICULUM

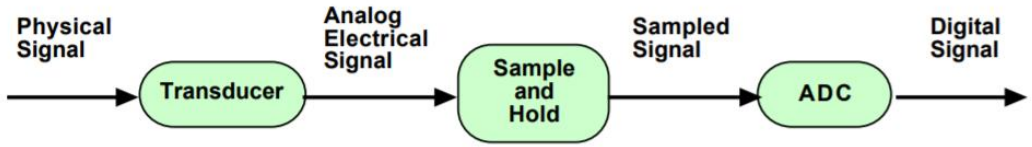


PROTOTYPING INTERACTIVE SYSTEMS

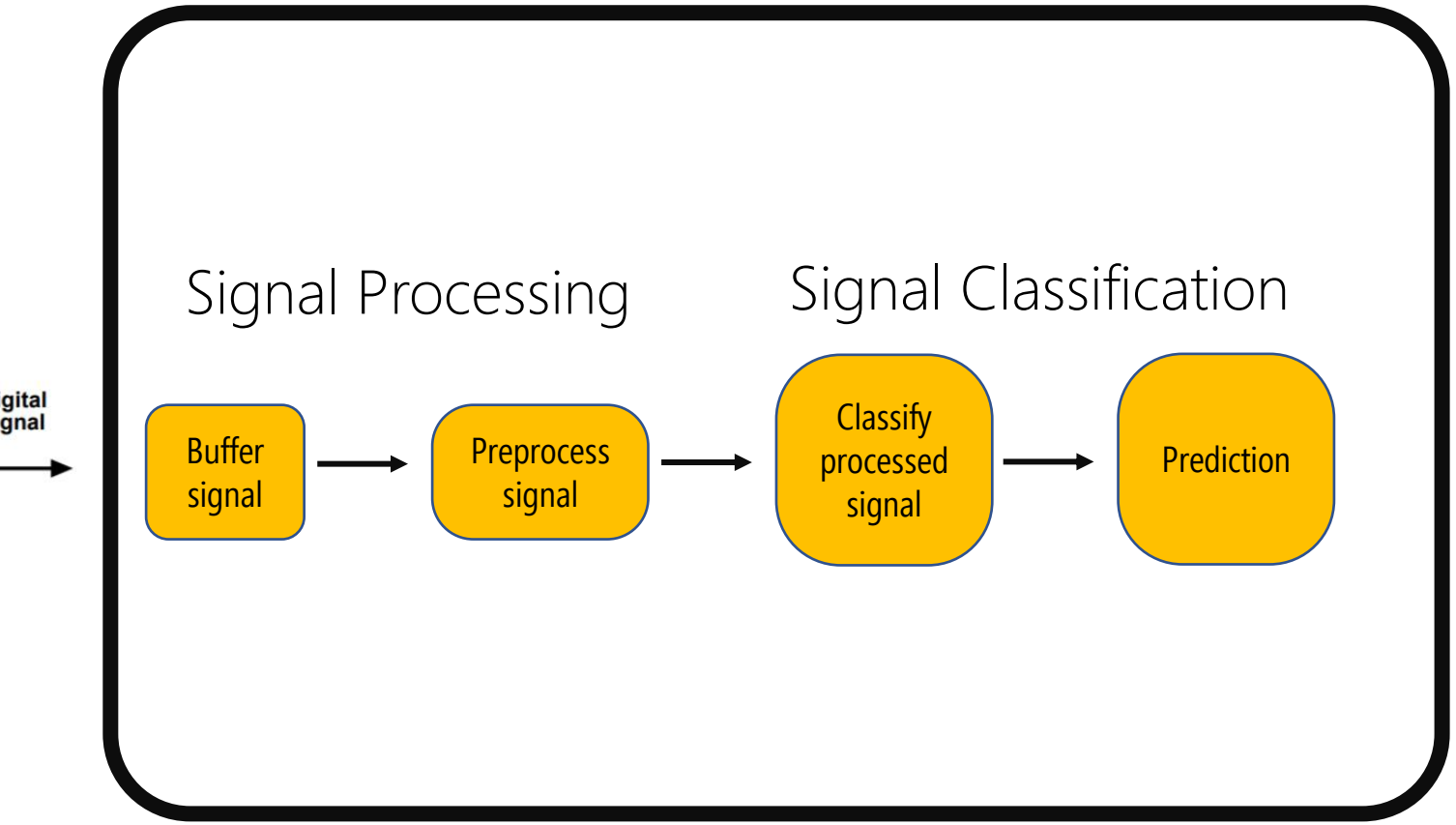
COURSE CURRICULUM



DATA FLOW FROM PHYSICAL SIGNAL TO PREDICTION



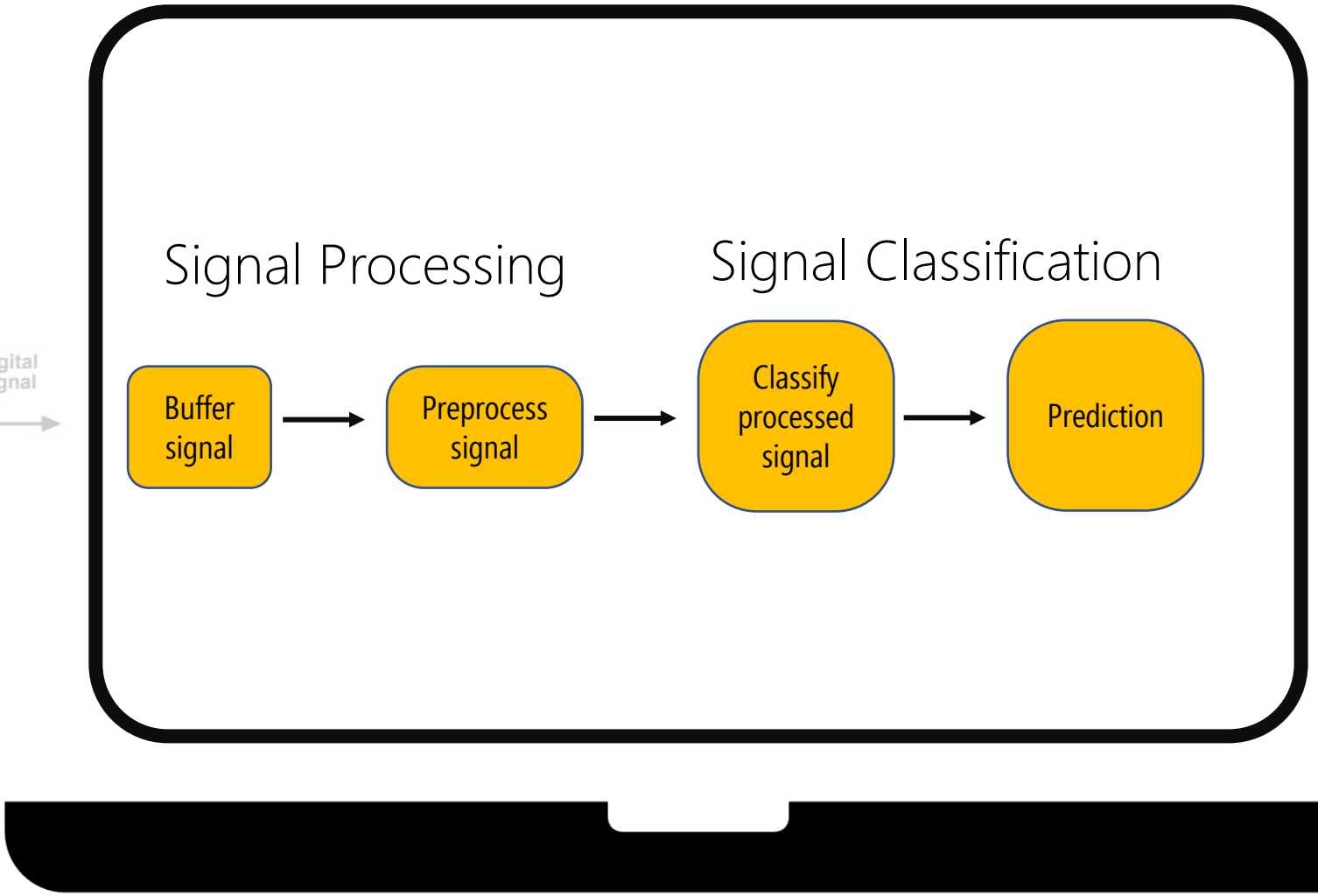
Signal Acquisition



Based on: Stiber, Stiber, & Larson, Signal Computing, 2016 (link)

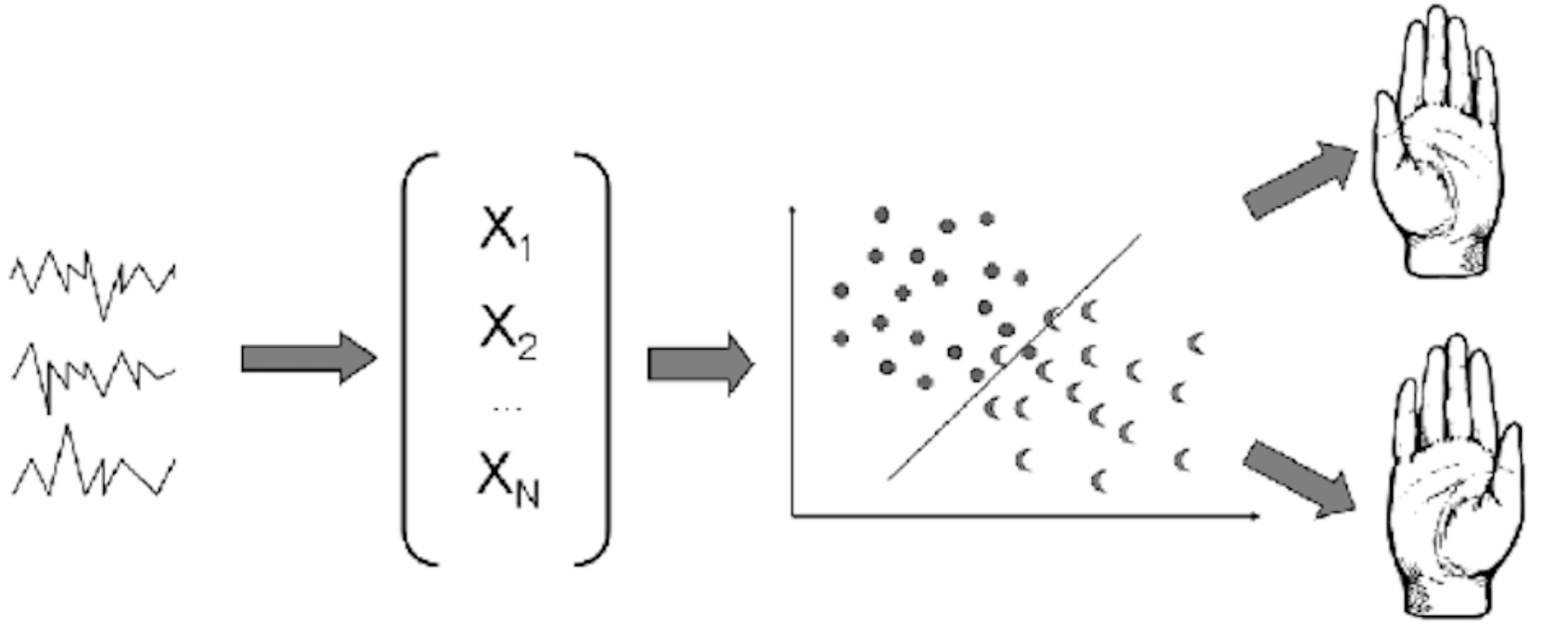
DATA FLOW FROM PHYSICAL SIGNAL TO PREDICTION

Signal Acquisition



Based on: Stiber, Stiber, & Larson, Signal Computing, 2016 (link)

PROTOTYPICAL ML PIPELINE



EEG signals

Ex: signal recorded during left or right hand motor imagery

Feature extraction

Ex: band power in the μ and β rhythms for electrodes located over the motor cortex

Classification

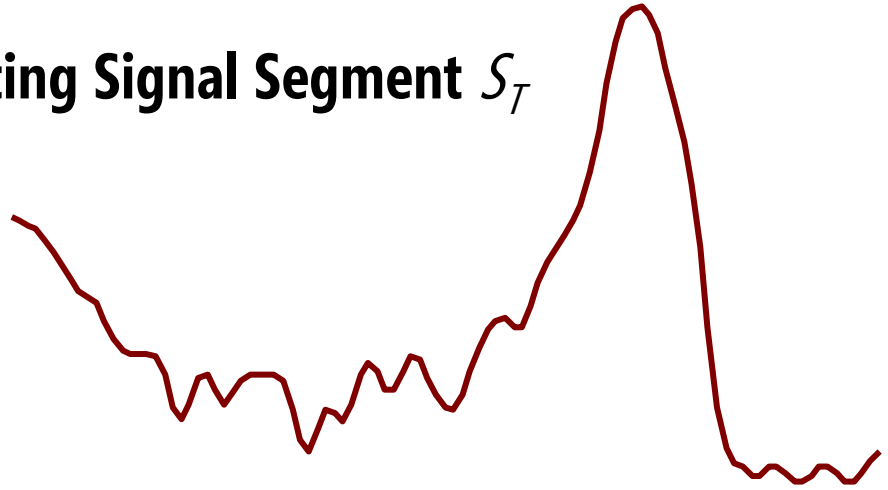
Ex: Linear Discriminant Analysis (LDA)

Estimated class

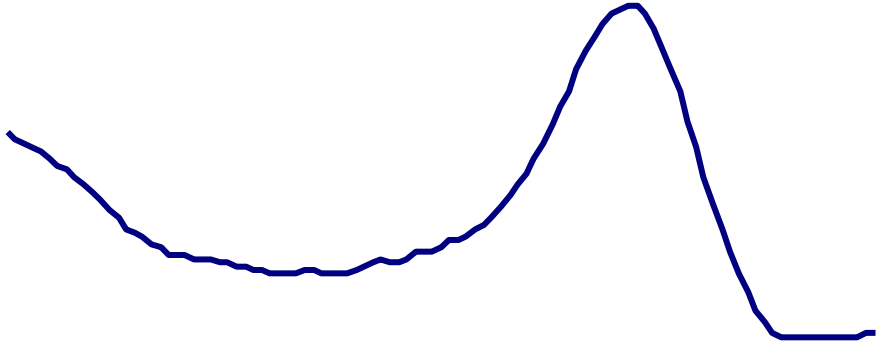
Ex: Left or Right (imagined hand movement)

CONSIDER TWO SIGNALS: IS S1 THE SAME AS S2?

Existing Signal Segment S_T



Incoming (New) Signal Segment S_N



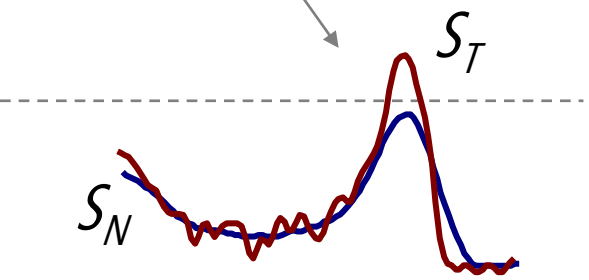
How can the computer determine whether S_N is the same signal as S_T?

INTRO TO ML

3 PREVAILING APPROACHES FOR SIGNAL CLASSIFICATION

1. RULE-BASED

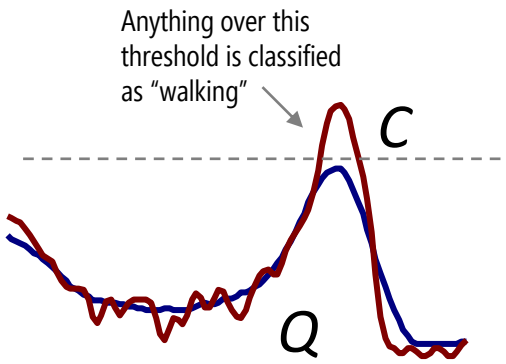
Anything over this threshold is classified as "walking." This threshold learned from "training" data



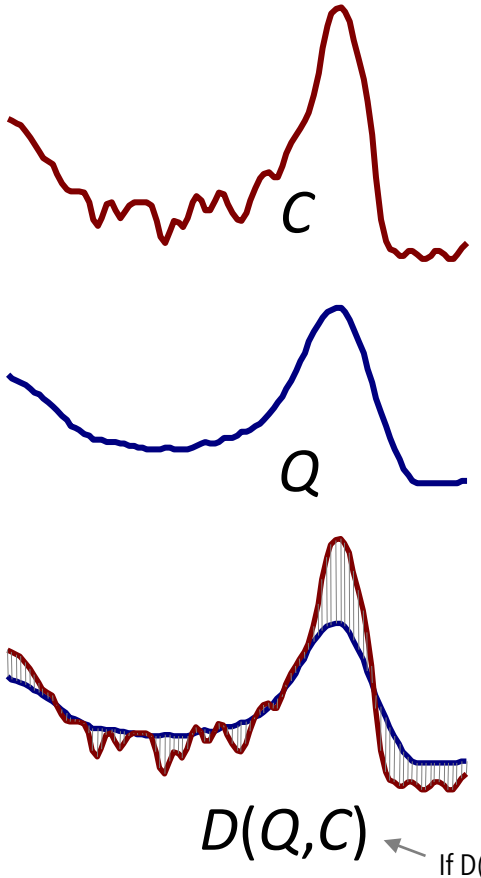
INTRO TO ML

3 PREVAILING APPROACHES FOR SIGNAL CLASSIFICATION

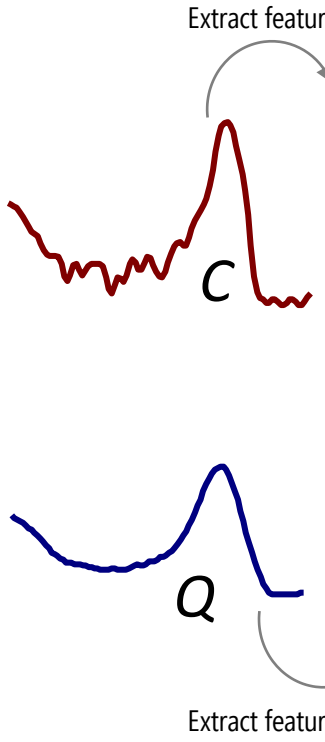
1. RULE-BASED



2. SHAPE MATCHING



3. FEATURE-BASED MODELS



Raw Signal	1st Derivative	FFT
Mean	Mean	Top N freqs
Variance	Variance	Fundamental
Skewness	Skewness	
Kurtosis	Kurtosis	

Raw Signal	1st Derivative	FFT
Mean	Mean	Top N freqs
Variance	Variance	Fundamental
Skewness	Skewness	
Kurtosis	Kurtosis	

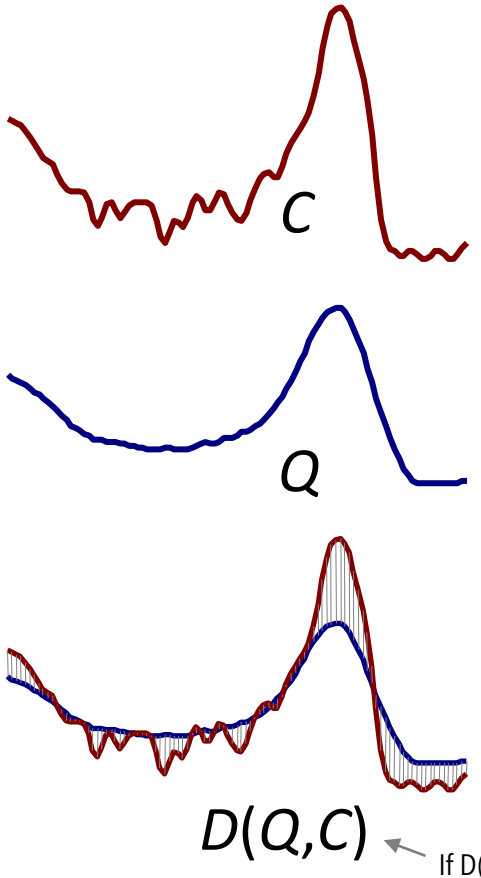
Supervised Learning Model

Output

3 PREVAILING APPROACHES FOR SIGNAL CLASSIFICATION

1. RULE-BASED

2. SHAPE MATCHING



If D(Q,C) < threshold, we consider them the same signal

3. FEATURE-BASED MODELS

Raw Signal	1st Derivative	FFT
Mean	Mean	Top N freqs
Variance	Variance	Fundamental
Skewness	Skewness	
Kurtosis	Kurtosis	

Raw Signal	1st Derivative	FFT
Mean	Mean	Top N freqs
Variance	Variance	Fundamental
Skewness	Skewness	
Kurtosis	Kurtosis	

Supervised Learning Model

Output

INTRO TO SHAPE-MATCHING

SIMILARITY METRICS

How do we formally define similarity?

Based on: Professor Eamonn Keogh's time-series analysis tutorials: http://www.cs.ucr.edu/~eamonn/tutorials.html

INTRO TO SHAPE-MATCHING

DEFINING DISTANCE MEASURES

Let O_1 and O_2 be two objects from the universe of possible objects.

The distance (dissimilarity) is denoted by $D(O_1,O_2)$

Based on: Professor Eamonn Keogh's time-series analysis tutorials: http://www.cs.ucr.edu/~eamonn/tutorials.html

DEFINING DISTANCE MEASURES

Let O_1 and O_2 be two objects from the universe of possible objects.

The distance (dissimilarity) is denoted by $D(O_1,O_2)$

Properties of a desirable distance metric:

D(A,B) = D(B,A)	Symmetry
D(A,A) = 0	Constancy
D(A,B) = 0 IIf A= B	Positivity

INTRO TO SHAPE-MATCHING

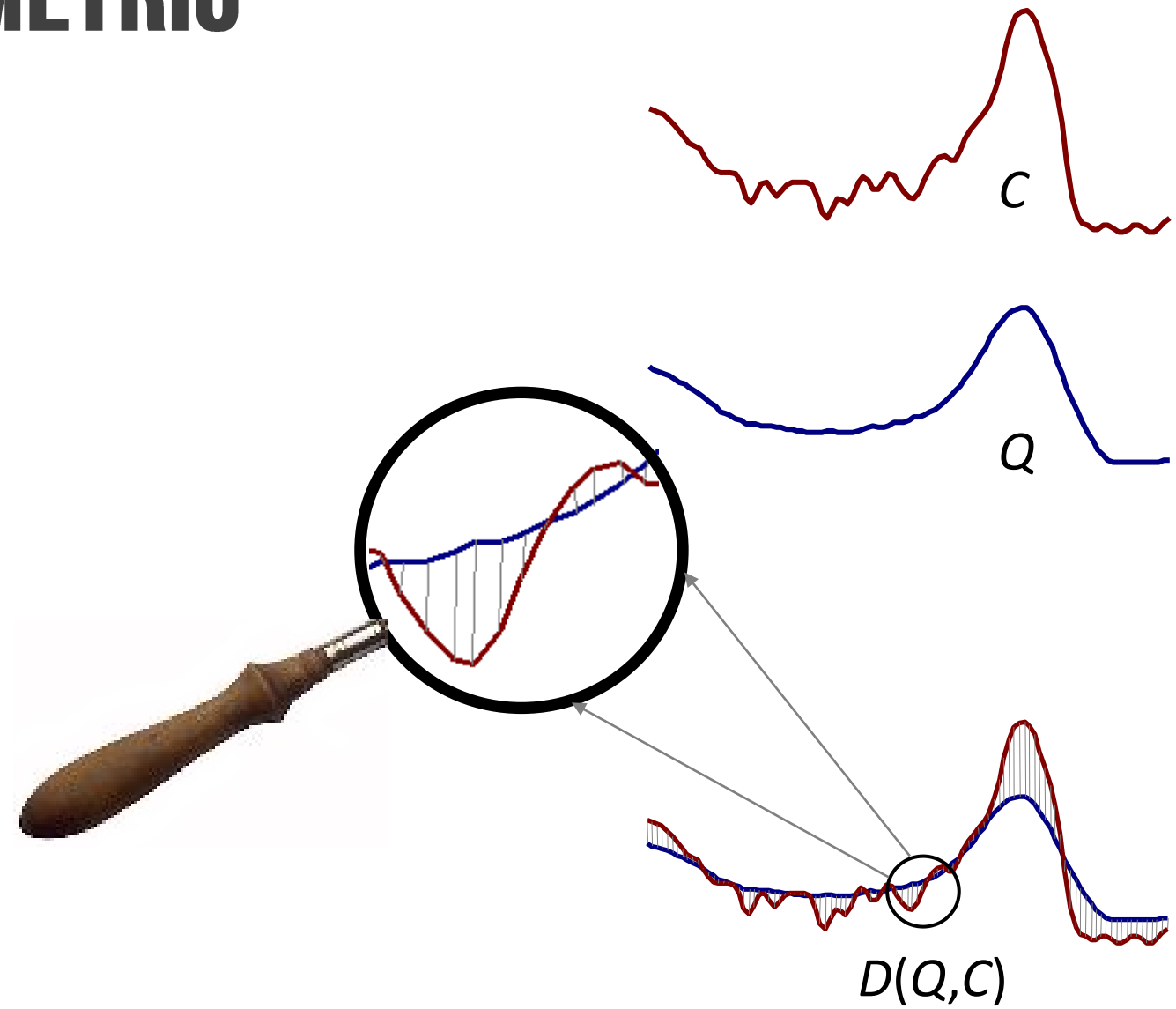
EUCLIDEAN DISTANCE METRIC

Given two time series:

$$Q = q_1 \ldots q_n$$

$$C = c_1 \ldots c_n$$

$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2}$$



Based on: Professor Eamonn Keogh's time-series analysis tutorials: http://www.cs.ucr.edu/~eamonn/tutorials.html

COMMON OPTIMIZATION OF EUCLID DISTANCE

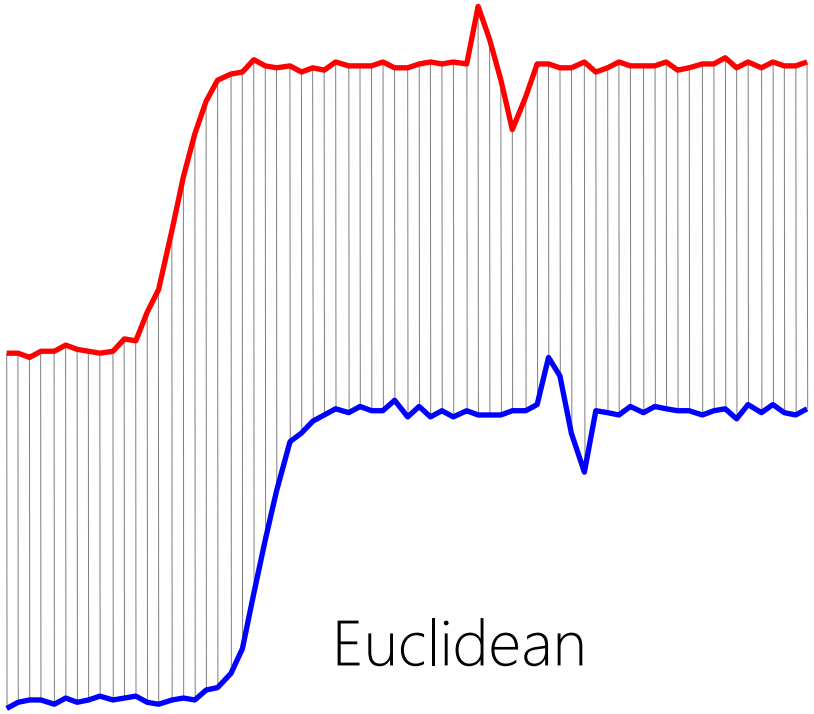
Instead of calculating the raw Euclidean distance, calculate the squared Euclidean distance (to save CPU time)

$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n}(q_i - c_i)^2} \quad \Rightarrow \quad D_{squared}(Q,C) \equiv \sum_{i=1}^{n}(q_i - c_i)^2$$

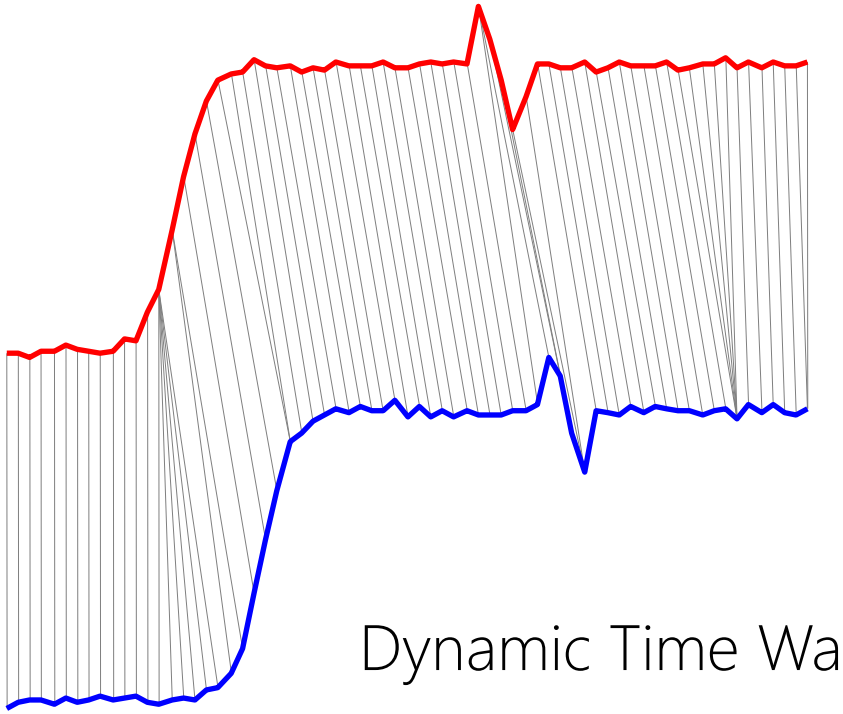
INTRO TO SHAPE-MATCHING

DYNAMIC TIME WARPING

Dynamic Time Warping is a more sophisticated similarity approach; however, it is $O(N^2)$



Euclidean



Dynamic Time Warping

Fixed Time Axis
Sequences are aligned "one-to-one"

"Warped" Time Axis
Non-linear alignments are possible

Based on: Professor Eamonn Keogh's time-series analysis tutorials: http://www.cs.ucr.edu/~eamonn/tutorials.html

PREPROCESSING TIME SERIES DATA

Shape-matching algorithms are particularly sensitive to distortions in the data that don't matter but greatly impact performance

Offset translation (*e.g.*, demeaning)

Amplitude scaling

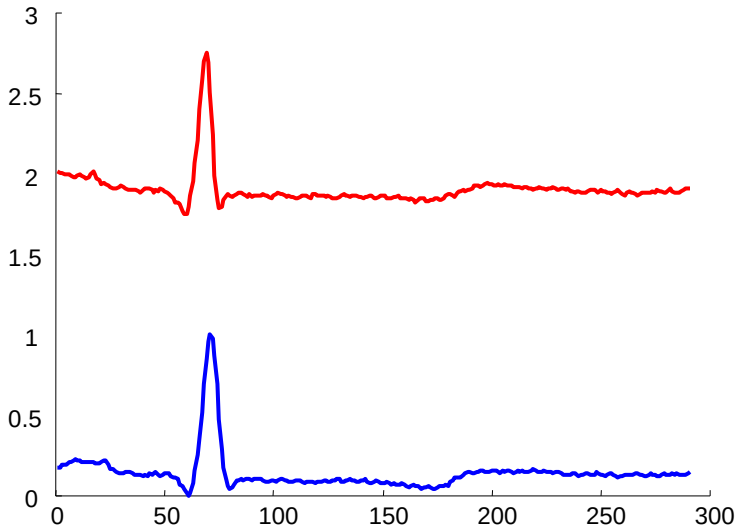
Detrending (*e.g.*, removing linear trend)

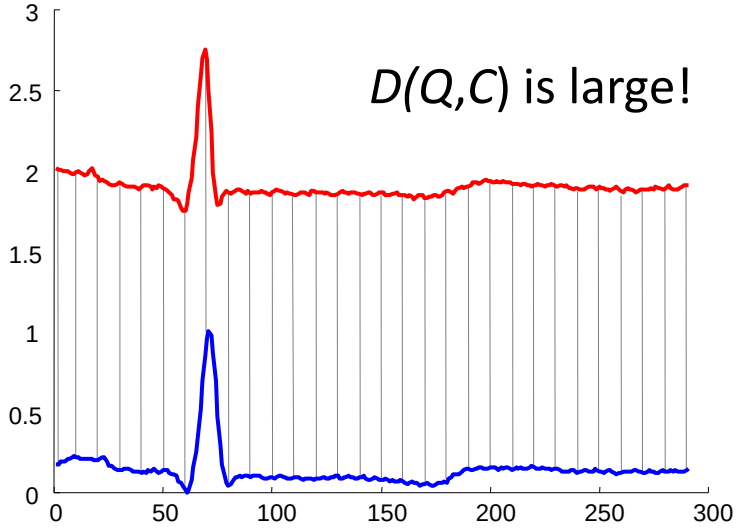
Removing noise

Based on: Professor Eamonn Keogh's time-series analysis tutorials: http://www.cs.ucr.edu/~eamonn/tutorials.html

PREPROCESSING TRANSFORMATIONS

OFFSET TRANSLATION

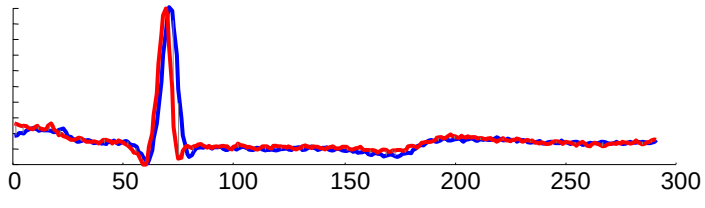




D(Q,C) is large!

$Q = Q - \text{mean}(Q)$

$C = C - \text{mean}(C)$

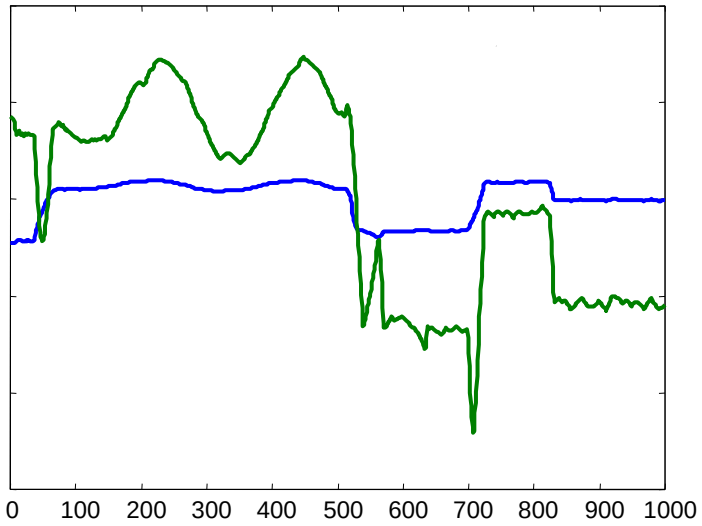


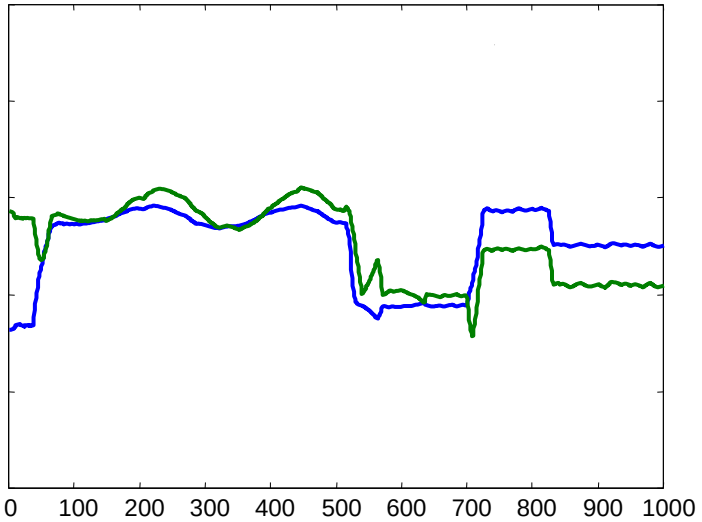
D(Q,C) is now small!

Based on: Professor Eamonn Keogh's time-series analysis tutorials: http://www.cs.ucr.edu/~eamonn/tutorials.html

PREPROCESSING TRANSFORMATIONS

AMPLITUDE SCALING





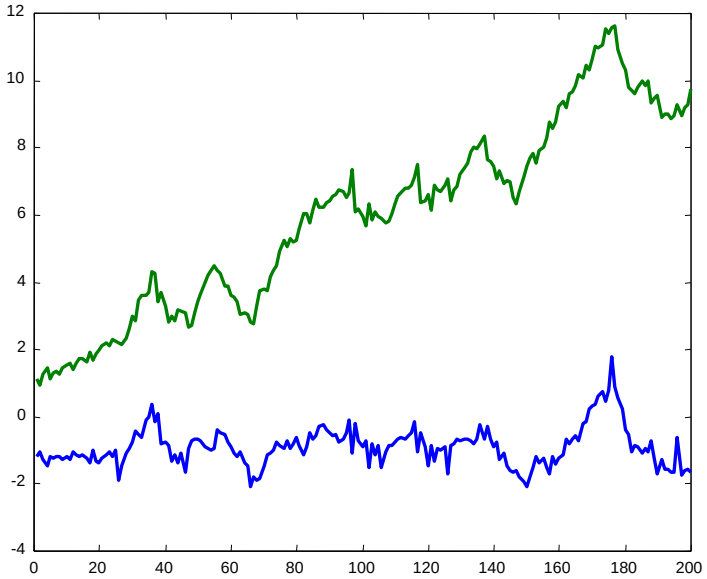
$Q = (Q - \text{mean}(Q)) / \text{std}(Q)$

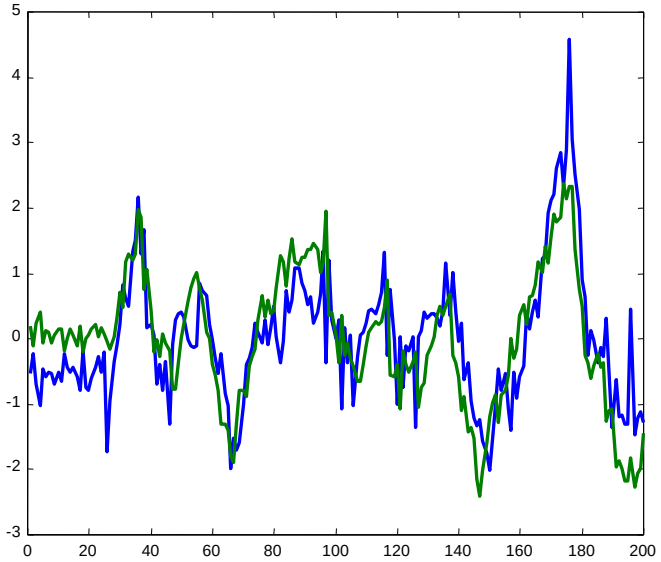
$C = (C - \text{mean}(C)) / \text{std}(C)$

Based on: Professor Eamonn Keogh's time-series analysis tutorials: http://www.cs.ucr.edu/~eamonn/tutorials.html

PREPROCESSING TRANSFORMATIONS

DETRENDING





Find best fit line to time series, then subtract that line from the signal

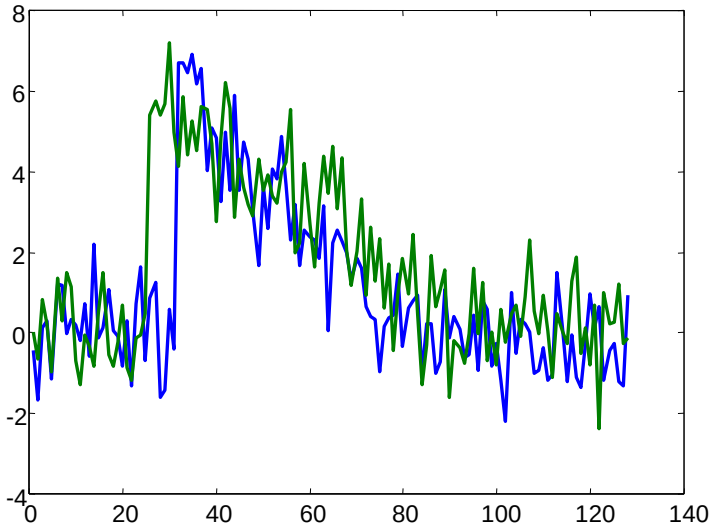
$Q = \text{detrend}(Q)$

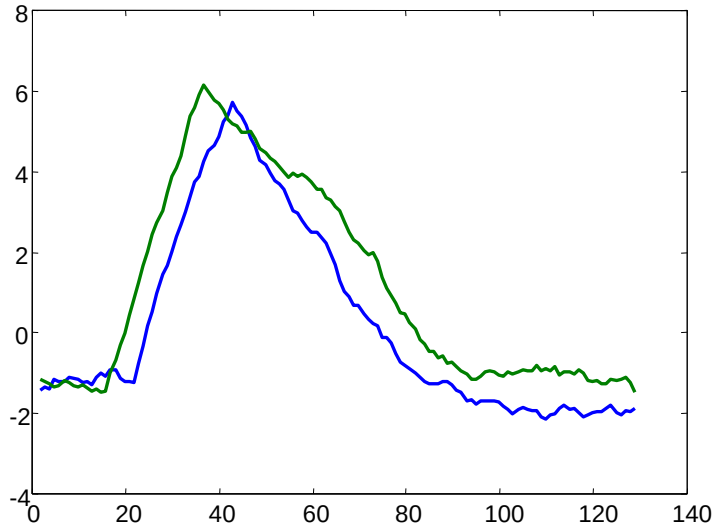
$C = \text{detrend}(C)$

Based on: Professor Eamonn Keogh's time-series analysis tutorials: http://www.cs.ucr.edu/~eamonn/tutorials.html

PREPROCESSING TRANSFORMATIONS

SMOOTHING





Q = smooth(Q)

C = smooth(C)

SHAPE MATCHING EXAMPLE

HYDROSENSE

HydroSense: Infrastructure-Mediated Single-Point Sensing of Whole-Home Water Activity

Jon Froehlich[1], Eric Larson[2], Tim Campbell[3], Conor Haggerty[4], James Fogarty[1], Shwetak N. Patel[1,2]

[1]Computer Science & Engineering, [2]Electrical Engineering,
[3]Mechanical Engineering, [4]Community, Environment, and Planning
DUB Institute, University of Washington
Seattle, WA 98195
{ jfroehli@cs, eclarson@u, tcampbll@u, conorh@u, jfogarty@cs, shwetak@cs } .washington.edu

ABSTRACT

Recent work has examined infrastructure-mediated sensing as a practical, low-cost, and unobtrusive approach to sensing human activity in the physical world. This approach is based on the idea that human activities (e.g., running a dishwasher, turning on a reading light, or walking through a doorway) can be sensed by their manifestations in an environment's existing infrastructures (e.g., a home's water, electrical, and HVAC infrastructures). This paper presents HydroSense, a low-cost and easily-installed single-point sensor of pressure within a home's water infrastructure. HydroSense supports both identification of activity at individual water fixtures within a home (e.g., a particular toilet, a kitchen sink, a particular shower) as well as estimation of the amount of water being used at each fixture. We evaluate our approach using data collected in ten homes. Our algorithms successfully identify fixture events with 97.9% aggregate accuracy and can estimate water usage with error rates that are comparable to empirical studies of traditional utility-supplied water meters. Our results both validate our approach and provide a basis for future improvements.

Author Keywords

Infrastructure-mediated sensing, activity sensing.

ACM Classification Keywords

H5.2. Information Interfaces and Presentation: User Interfaces; H1.2. Models and Principles: User/Machine Systems.

General Terms

Algorithms, Experimentation, Measurement

INTRODUCTION AND MOTIVATION

Effective methods for sensing and modeling human activity in the physical world are a cornerstone of ubiquitous computing research and practice. Many complementary approaches have been developed, including recent interest in *infrastructure-mediated sensing* [7, 14, 15, 16, 17, 20]. This approach is based on the idea that human activities (e.g., running a dishwasher, turning on a reading light, or walking through a doorway) can be sensed via their manifestations in an environment's existing infrastructures (a home's water [7], electrical [15, 16, 17, 20], and HVAC [14] infrastructures). Because of its practical, low-cost, and unobtrusive nature, infrastructure-mediated sensing offers significant promise as a general method.

This work focuses on infrastructure-mediated sensing of home water activity. Water is essential to many home activities (e.g., washing, cleaning, cooking, drinking, gardening) which are in turn central to important potential ubiquitous computing applications (e.g., helping elders live more independently, helping people monitor their own water usage to reduce waste). Previous work monitoring home water usage [6, 7, 9] required multiple sensing points, exposed piping, could not infer *both* fixture and flow, and received limited or no validation in actual homes.

This paper presents HydroSense, a low-cost, *single-point* solution for activity sensing mediated by a home's existing water infrastructure. HydroSense is based on continuous analysis of *pressure* within a home's water infrastructure. Specifically, we *identify individual water fixtures* (e.g., a particular toilet, a kitchen sink, a particular shower) within a home according to the unique *pressure waves* that propagate to the sensor when valves are opened or closed. We also estimate the *amount of water being used* at a fixture based on the magnitude of the resulting *pressure drop* within the water infrastructure. Our work represents a significant advance over prior research in several regards:

First, HydroSense can be *easily installed* at any accessible location within a home's existing water infrastructure. Typical installations will be at an exterior hose bib, utility sink spigot, or water heater drain valve. If unavailable or not easily accessed (e.g., in an apartment unit), HydroSense can also be installed at the water connection point for a dishwasher, clothes washer, or toilet. All of these are simple screw-on installation points, with no need for a plumber.

Second, HydroSense's analysis of *pressure* provides the unique capability of sensing both the *individual fixture* at which water is currently being used as well as an estimate of the *amount of water being used*. HydroSense is the first practical approach to enabling applications that require both. Our sensing of pressure is also less susceptible to ambient noise, as has been encountered in previous microphone-based infrastructure-mediated systems.

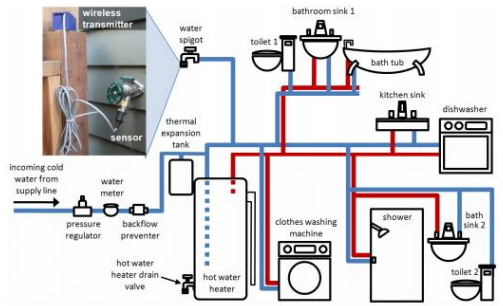


Figure 1: An illustrative schematic of a basic plumbing layout in a two-bathroom home. HydroSense can be easily installed at any accessible location in a home's water infrastructure, with typical installations at an exterior hose bib (shown above), a utility sink spigot, or a water heater drain valve. By continuously sensing water *pressure* at this single installation point, HydroSense can both *identify individual fixtures* at which water is being used as well as estimate the *amount of water being used*.

Third, we evaluate HydroSense in ten very diverse homes, thus providing a more *robust evaluation* than any previous work on water-related home activity sensing. We demonstrate reliable segmentation of valve pressure *events* from the surrounding sensor stream, show reliable classification of *valve open* and *valve close* events, show the successful identification of *individual fixtures* with 97.9% aggregate accuracy, and show that an appropriately located and calibrated system can estimate water usage with error rates comparable to empirical studies of traditional utility-supplied water meters. In addition, we present initial forward-looking analyses of compound event detection, a comparison of sensing at different locations, and a first look at the temporal stability of pressure event signatures. Our evaluation both validates the feasibility of our approach and provides a basis for future analyses and improvements.

Figure 1 illustrates a typical plumbing arrangement in a two-bathroom home (discussed in greater detail in a later section). Figure 2 shows an actual annotated signal captured by our sensor. The signal is a kitchen faucet fixture being turned on, captured by our sensor at an exterior water bib. The remainder of this paper first discusses the theory behind our approach, presents our sensor implementation, and summarizes our in-home data collections. We then present our analyses of individual fixture identification and water flow estimation, follow by a discussion of some important directions for future work.

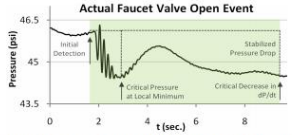


Figure 2: An actual event generated when a kitchen faucet is turned on, detected by our sensor at an exterior water bib. Our fixture identification is based in segmenting the event (indicated by the highlight) and then classifying it according to its shape (see also Figure 3 and Figure 6).

RELATED WORK

Prior work has demonstrated at least three approaches to the fundamental challenge of sensing human activity in the physical world: mobile and wearable sensing, distributed direct environmental sensing, and infrastructure-mediated environmental sensing. Promising mobile and wearable activity sensing methods include accelerometer-based activity recognition [2, 10] and the detection of interaction with tagged objects via a wearable RFID reader [18]. There are many compelling applications of mobile and wearable methods, but they share a common need for a person to be willing to wear or carry the necessary device.

Environmental sensing systems take a complementary perspective, instrumenting an environment to detect activity within it. In the home, distributed direct sensing can be

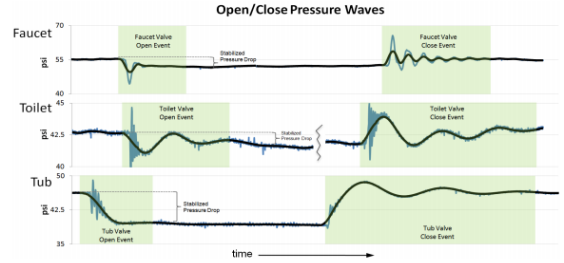


Figure 6: Several actual sensor streams from our in-home data collections. Each stream corresponds to a water valve being opened, remaining open for some amount of time, and then closed. We separately segment the valve open and valve close events from the sensor stream, as indicated by the highlighted regions of the streams. We estimate water flow to the valve based on the stabilized pressure drop while the valve remains open (the difference in pressure before the valve open versus after it).

ANALYSIS OF FIXTURE EVENT IDENTIFICATION

Given our collected data, we now pursue a three-step approach to examine the feasibility of identifying individual fixture events according to the unique transient pressure waves that propagate to our sensor. Recall that each *valve event* corresponds to water hammer when a valve is either opened or closed. We first *segment* each individual valve event from the stream, identifying its beginning and end to enable further analysis. We then *classify* each valve event as either a *valve open* or a *valve close* event. Finally, we *classify* the valve event according to the *individual fixture* that generated it. This section explicitly considers only events that occur in isolation, deferring our discussion and analysis of compound events until a later section.

Valve Event Segmentation

Before analyzing the characteristics of a valve event, we first segment it (i.e., isolate it) from the surrounding sensor stream. Segmentation must be effective for many different types of events, and so it is important to consider only features that are likely to be most typical of all valve events. Our approach is illustrated in Figure 6 (and in Figure 2 and Figure 3). The raw signal is smoothed using a low-pass linear phase finite impulse response filter. The smoothed signal and its derivative are then analyzed in a sliding window of 1000 samples (one second of sensed pressure).

The beginning of a valve event corresponds to one of two conditions. The most common is when the derivative of the smoothed signal exceeds a specified threshold relative to static pressure, indicating a rapid change (approximately 2 psi/sec for a home with 45 psi static pressure, scaled by the home's actual static pressure). The less common second condition is when the difference between the maximum and minimum values in the sliding window exceeds a threshold relative to static pressure, indicating a slow but substantial change (approximately 1 psi for a home with 45 psi static pressure, scaled by the actual static pressure). After the beginning of a valve event is detected via either method, the next change in the sign of the derivative represents the extreme of this valve event relative to the preceding static pressure (which may be a maximum or a minimum).

The end of a segmented valve event is typically detected as the first point at which an extreme of a fluctuation (a change in the sign of the derivative, dP/dt) is less than 5% of the magnitude of the first extreme following the beginning of the event. It is also possible for an event to be ended by a rapid increase in the magnitude of fluctuation. This corresponds to the occurrence of a compound event, as we will discuss in greater detail in a later section.

Applying this method to our collected in-home data yielded appropriate segmentations of 100% of our valve events from their surrounding sensor stream.

Classifying Valve Open and Valve Close Events

After segmenting each valve event, we classify it as either a *valve open* or a *valve close* event. We apply a hierarchical classifier that first considers the difference in the smoothed pressure at the beginning and the end of the segmented event. If the magnitude of this difference exceeds a threshold (approximately 2 psi for a home with 45 psi static pressure, scaled by the actual static pressure), the event can be immediately classified (a pressure decrease corresponds to a *valve open* and a pressure increase to a *valve close*). Otherwise, the event is classified according to the average value of the derivative between its beginning and its first extreme. A *valve open* creates an initial pressure decrease (a positive average derivative), while *valve close* events create an initial pressure increase (a negative average derivative).

Applying this method to the segmented valve events from our collected in-home data yields 100% correct classification of *valve open* and *valve close* events.

Source: Froehlich *et al.*, UbiComp 2009; Froehlich *et al.*, Pervasive 2011

How can we sense water usage
at every fixture in your home?

HYDROSENSE

DIRECT SENSING APPROACH

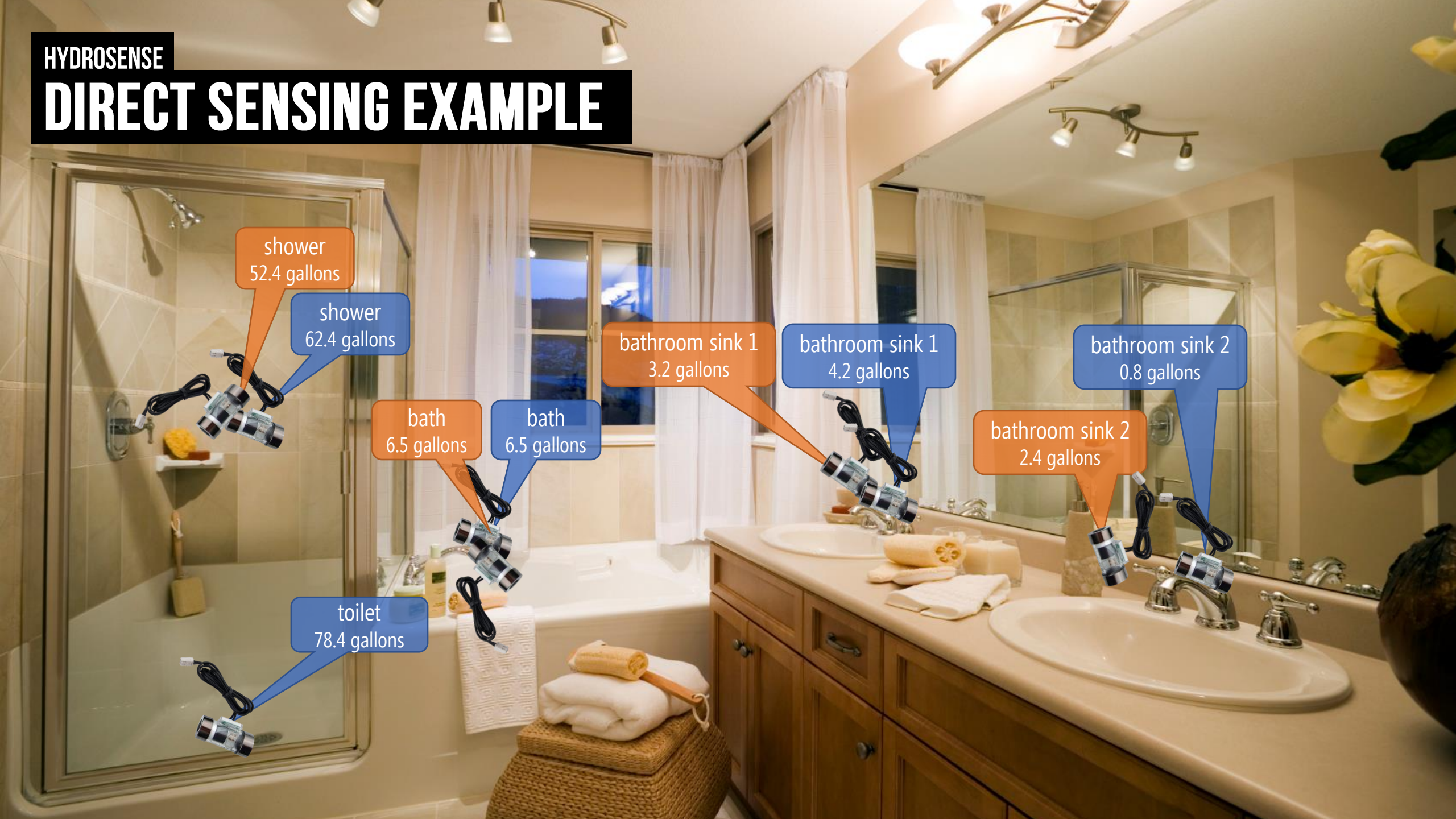
Source: Teague Labs, Arduino Water Meter, http://labs.teague.com/?p=722

HYDROSENSE

DIRECT SENSING EXAMPLE

HYDROSENSE

DIRECT SENSING EXAMPLE



HYDROSENSE

DIRECT SENSING EXAMPLE

shower
52.4 gallons

shower
62.4 gallons

bath
6.5 gallons

bath
6.5 gallons

toilet
78.4 gallons

bathroom sink 1
3.2 gallons

bathroom sink 1
4.2 gallons

bathroom sink 2
2.4 gallons

bathroom sink 2
0.8 gallons

HYDROSENSE

INDIRECT SENSING APPROACH

Source: Froehlich *et al.*, UbiComp'09; Froehlich et al., Pervasive'11; Larson *et al.*, Pervasive and Mobile Computing'12

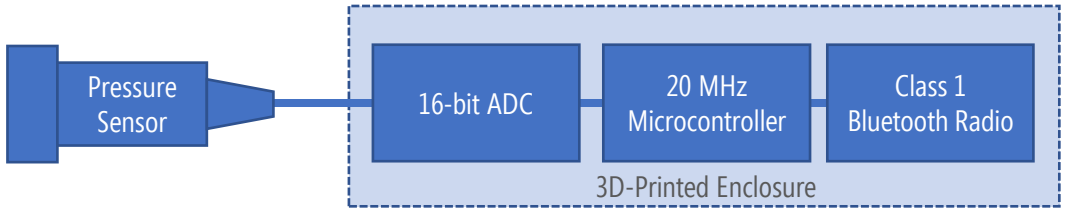
HYDROSENSE

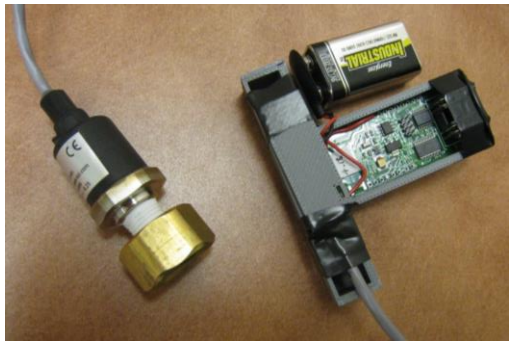
INDIRECT SENSING APPROACH

HydroSense attempts to infer fixture-level usage for the entire home from a **single** point.

Source: Froehlich *et al.*, UbiComp'09; Froehlich et al., Pervasive'11; Larson *et al.*, Pervasive and Mobile Computing'12

HYDROSENSE HW: PRESSURE-SENSING APPROACH



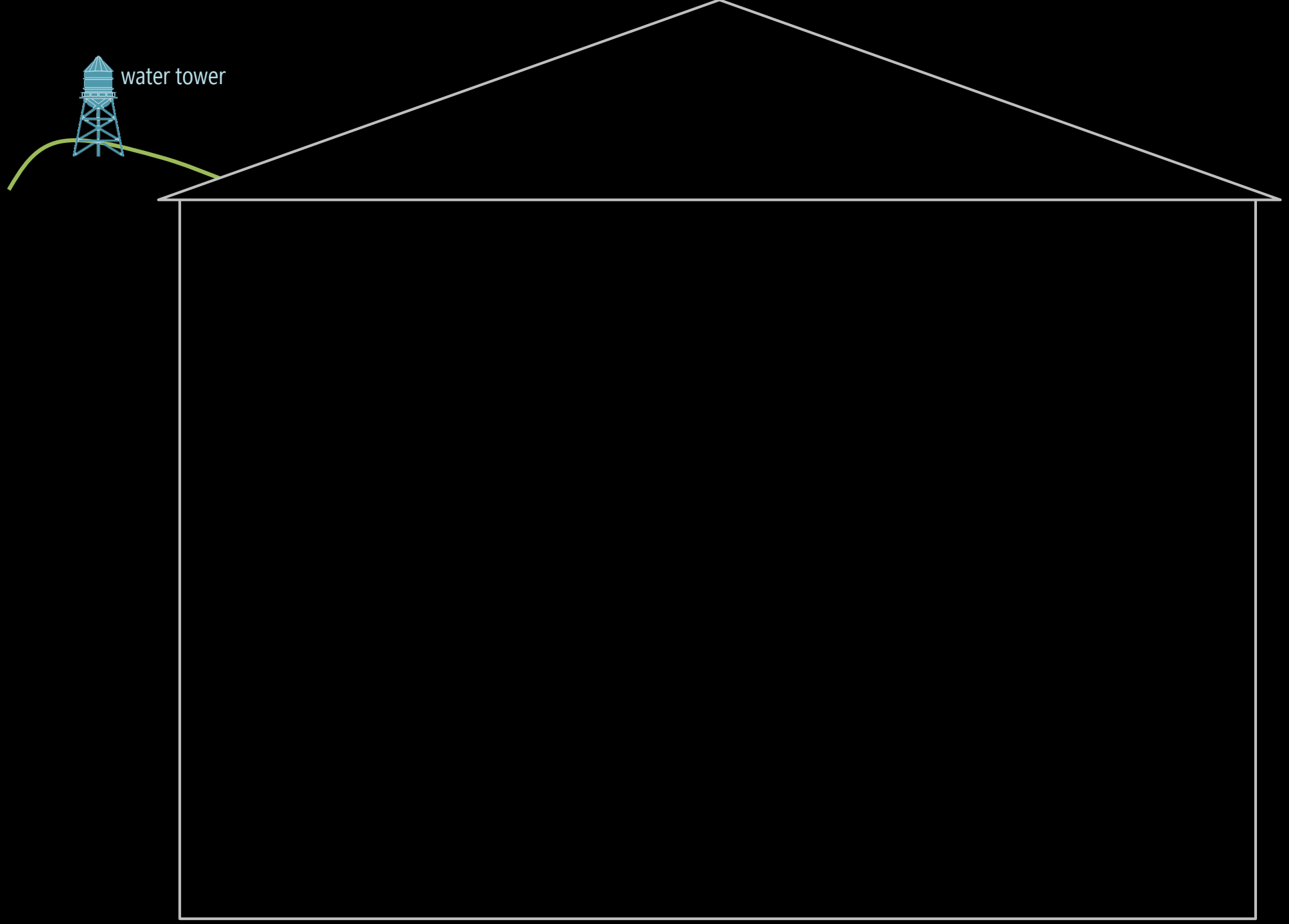


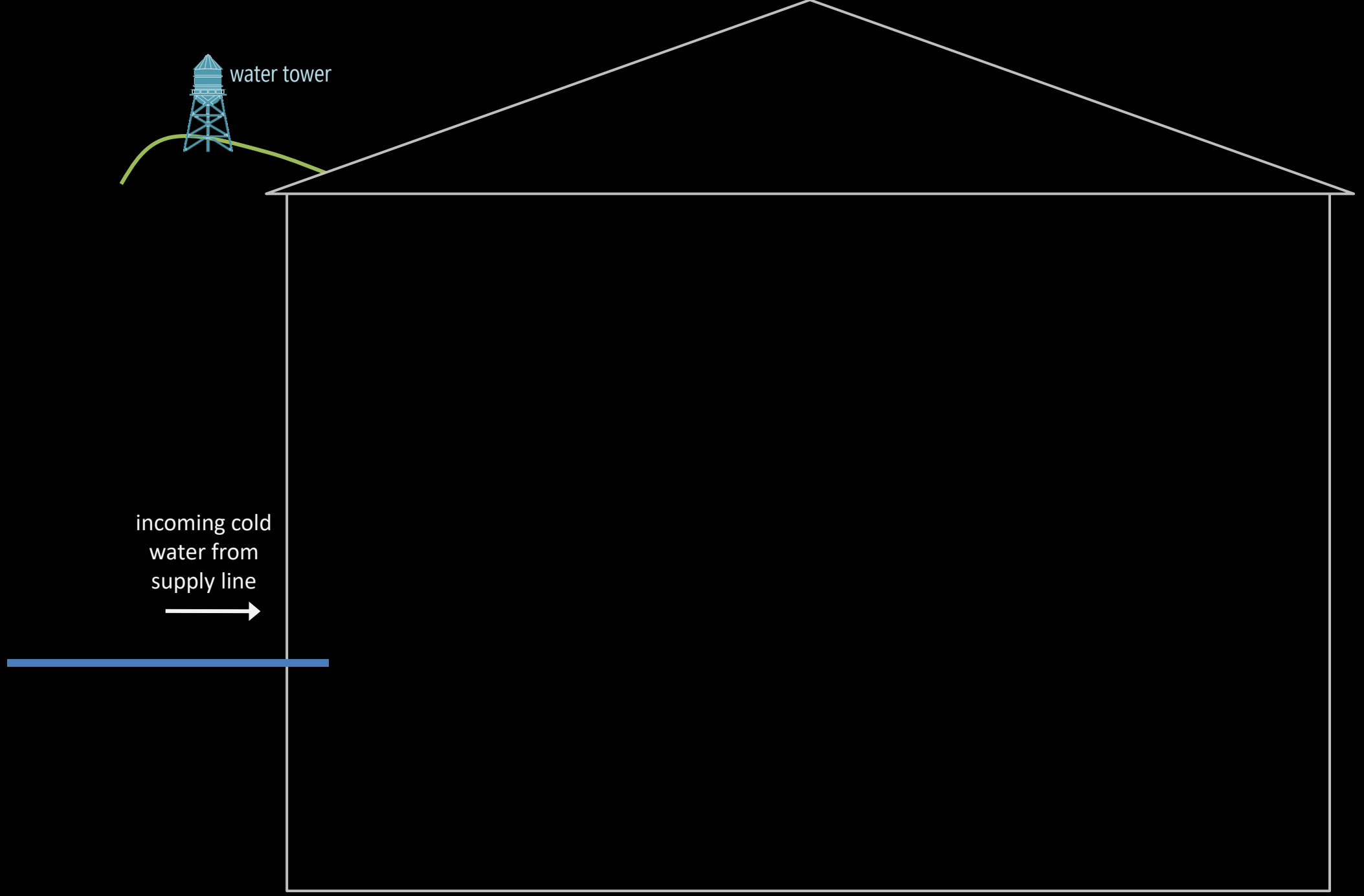
Source: Froehlich *et al.*, UbiComp'09; Froehlich et al., Pervasive'11; Larson *et al.*, Pervasive and Mobile Computing'12

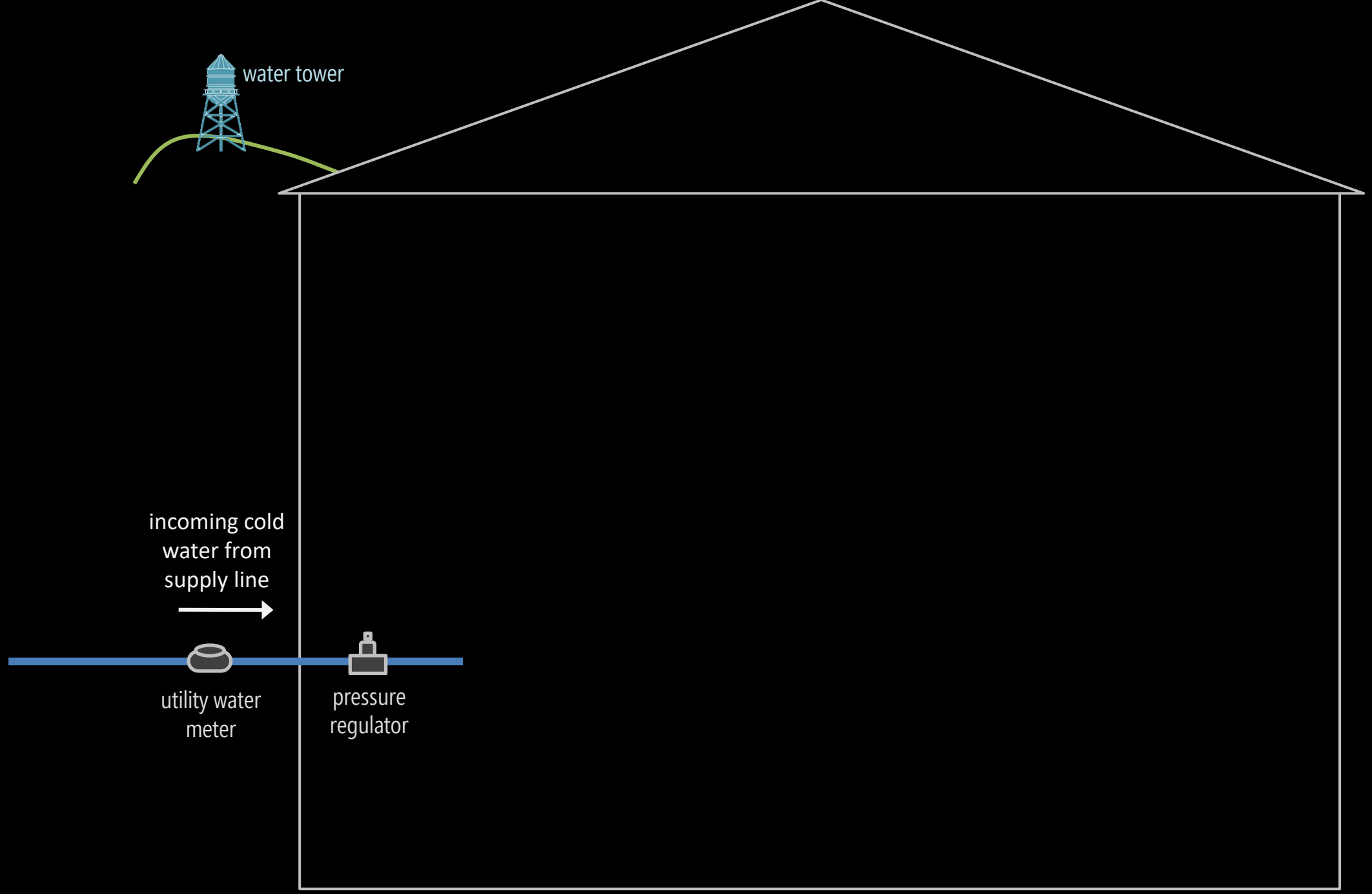
How does it work?

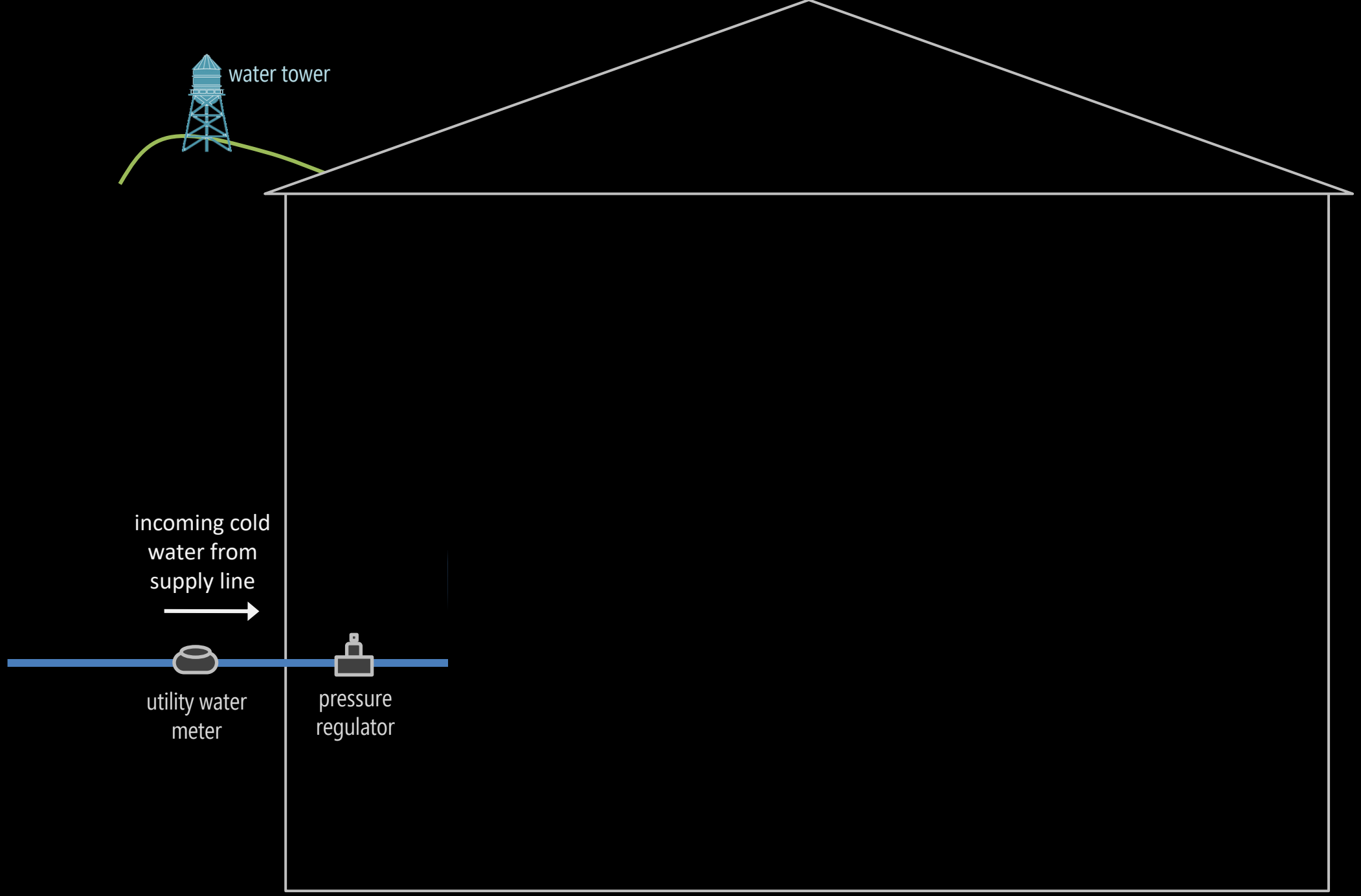
HYDROSENSE

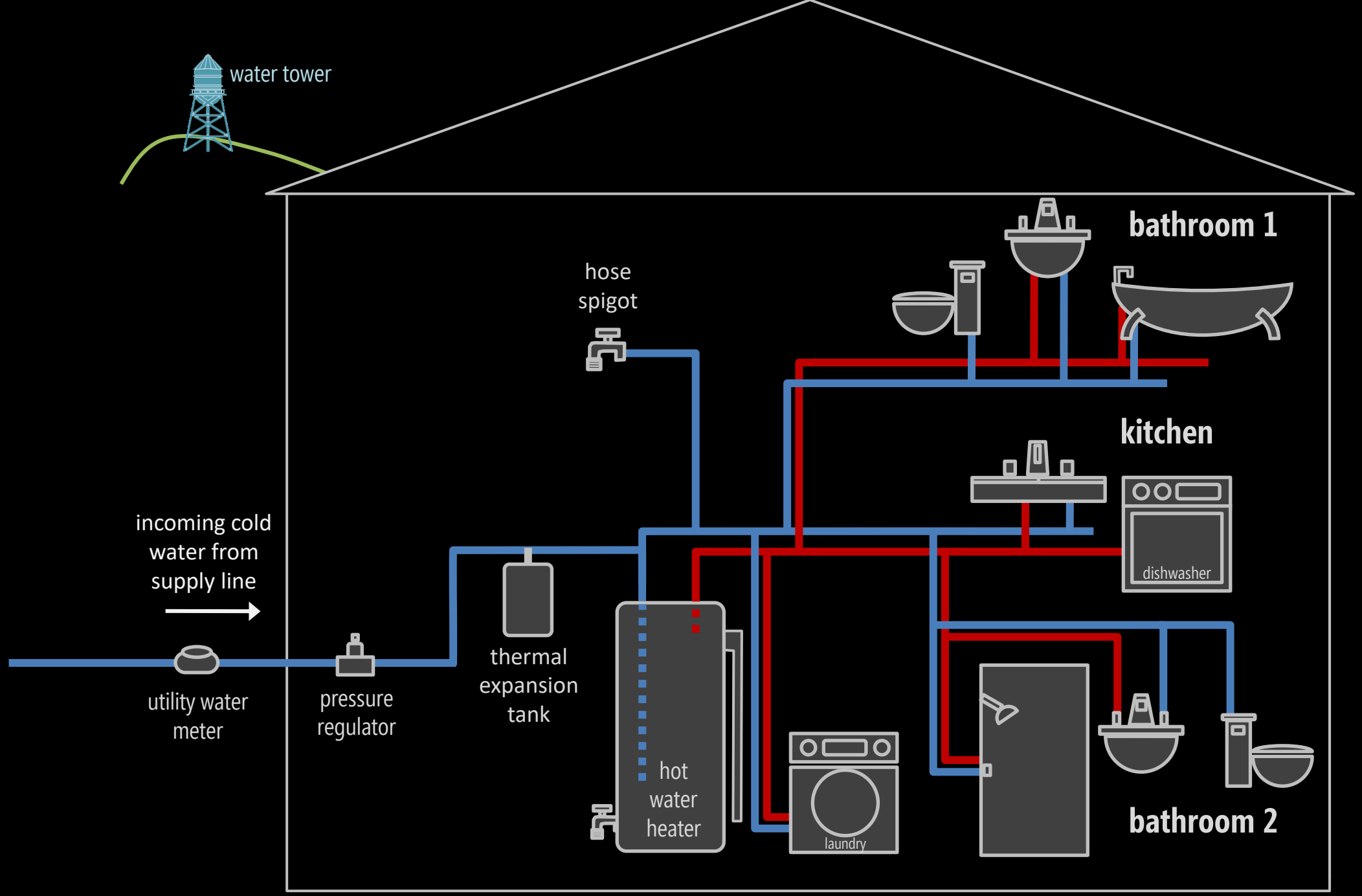
BRIEF PLUMBING TUTORIAL

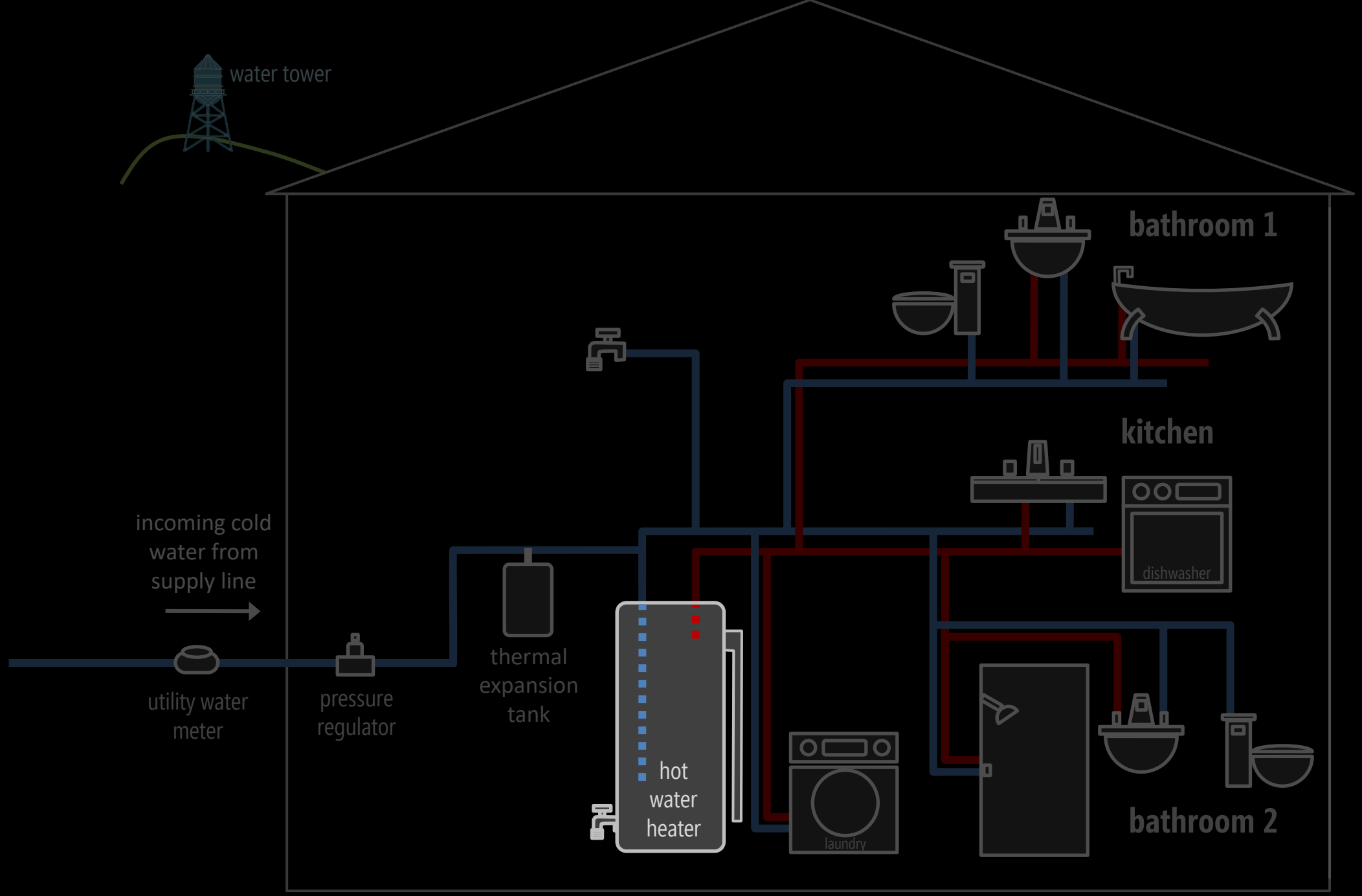

water tower

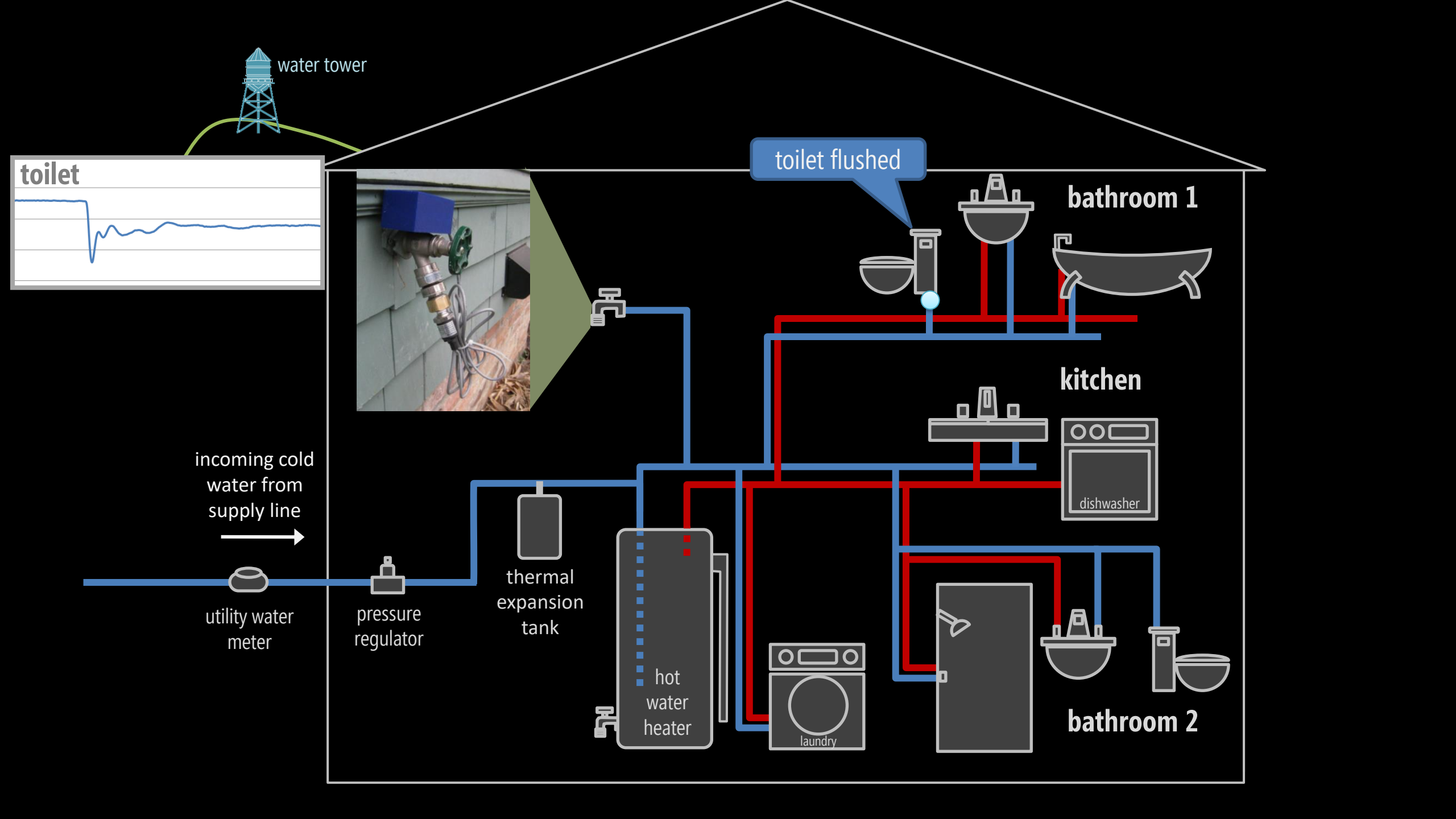

water tower
incoming cold water from supply line

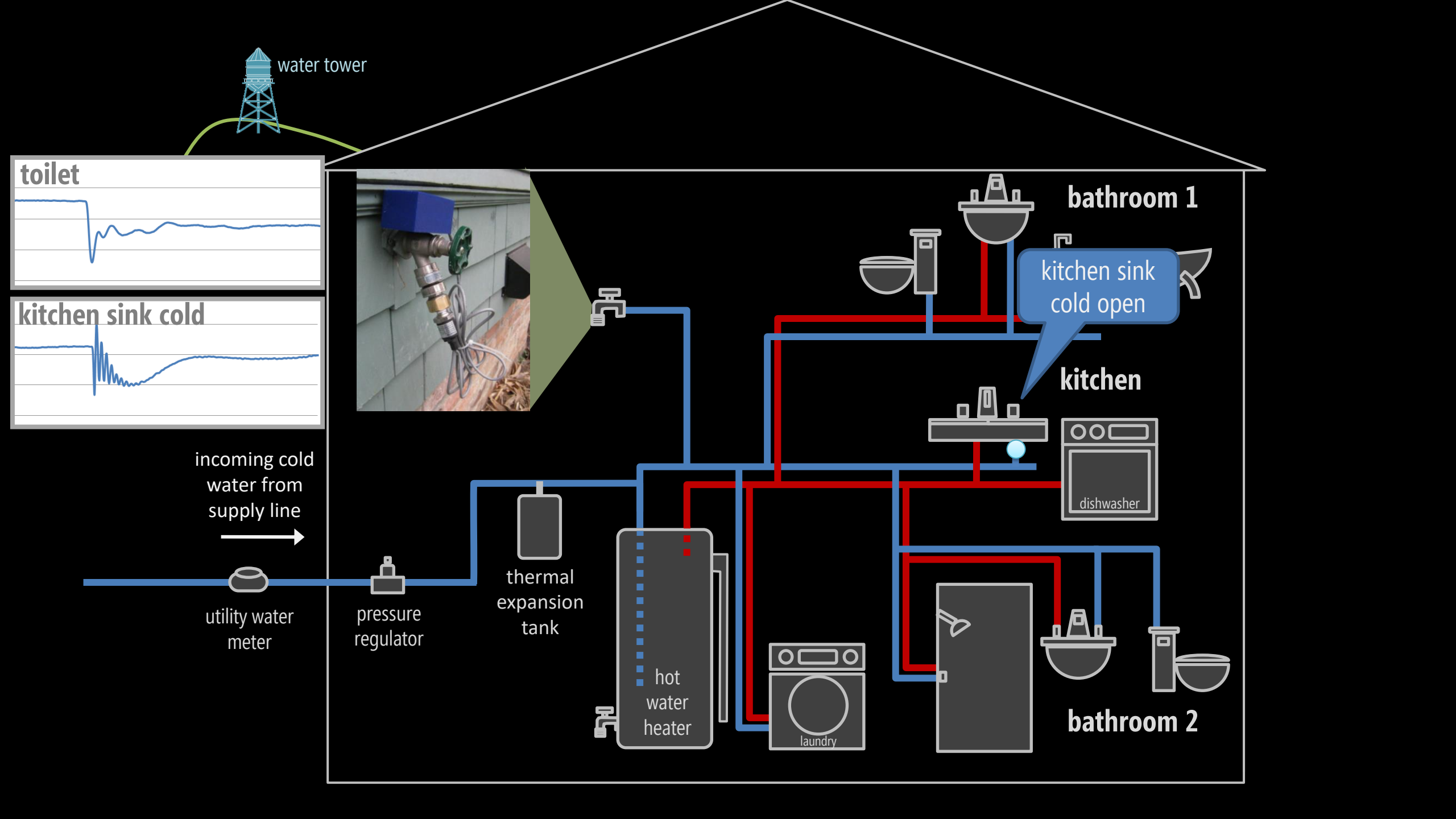

water tower
incoming cold water from supply line
utility water meter
pressure regulator

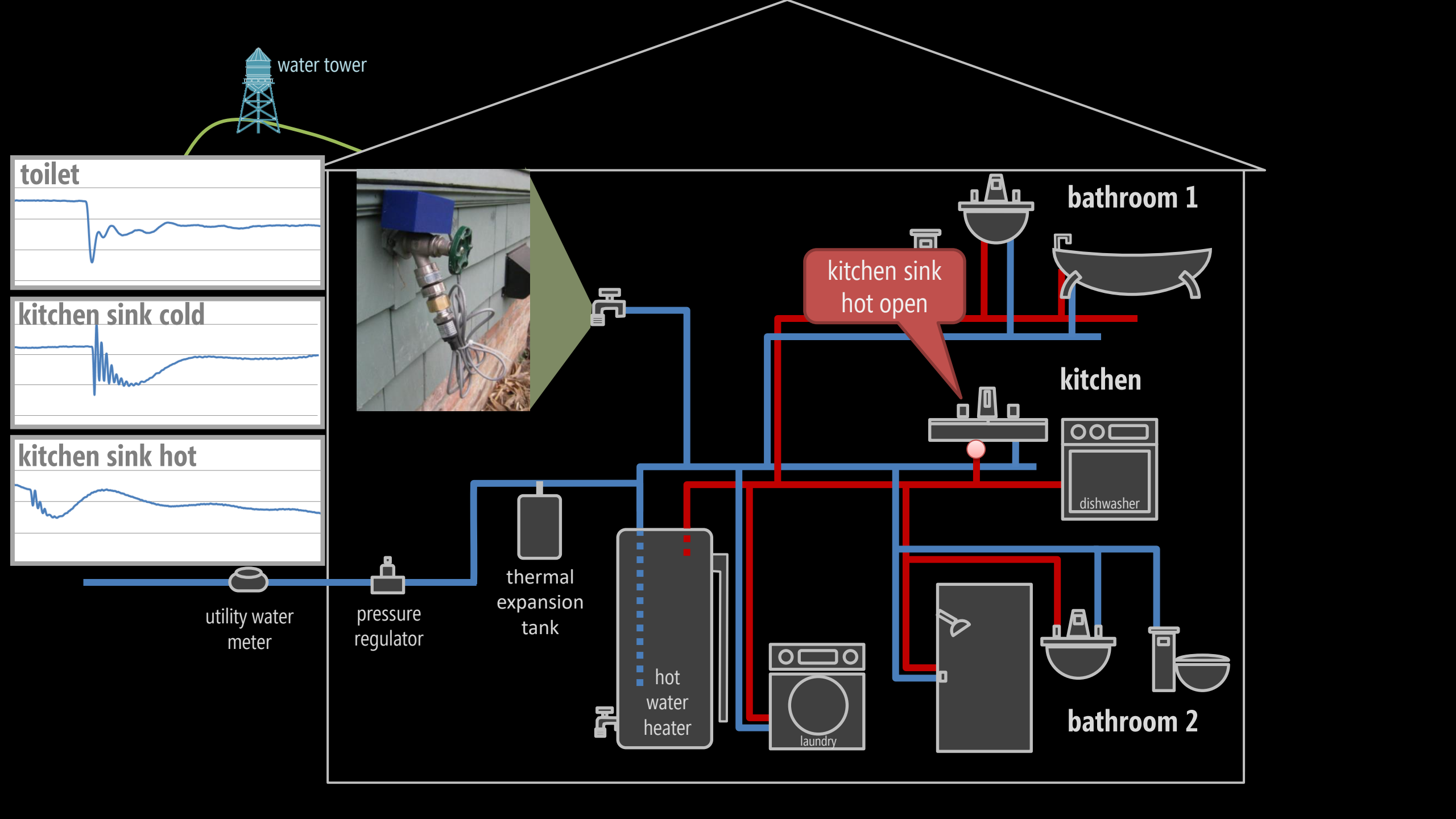

water tower
incoming cold water from supply line
utility water meter
pressure regulator

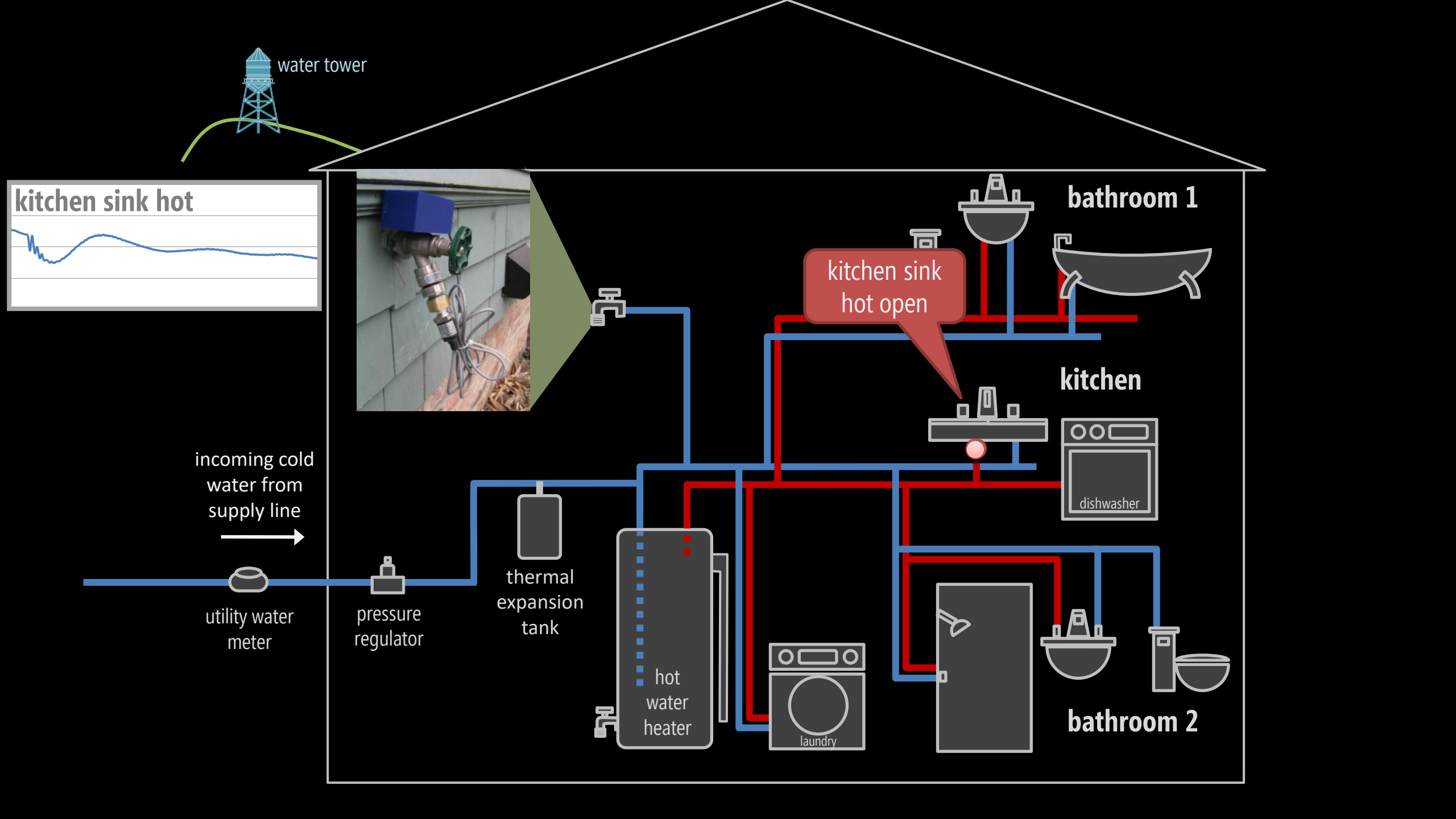

water tower
bathroom 1
hose spigot
kitchen
dishwasher
incoming cold water from supply line
thermal expansion tank
utility water meter
pressure regulator
hot water heater
laundry
bathroom 2


water tower
bathroom 1
kitchen
dishwasher
incoming cold water from supply line
utility water meter
pressure regulator
thermal expansion tank
hot water heater
laundry
bathroom 2


water tower
toilet
toilet flushed
bathroom 1
kitchen
dishwasher
incoming cold water from supply line
utility water meter
pressure regulator
thermal expansion tank
hot water heater
laundry
bathroom 2


water tower
toilet
kitchen sink cold
incoming cold water from supply line
utility water meter
pressure regulator
thermal expansion tank
hot water heater
laundry
bathroom 1
kitchen sink cold open
kitchen
dishwasher
bathroom 2


water tower
toilet
kitchen sink cold
kitchen sink hot
kitchen sink hot open
bathroom 1
kitchen
dishwasher
thermal expansion tank
utility water meter
pressure regulator
hot water heater
laundry
bathroom 2

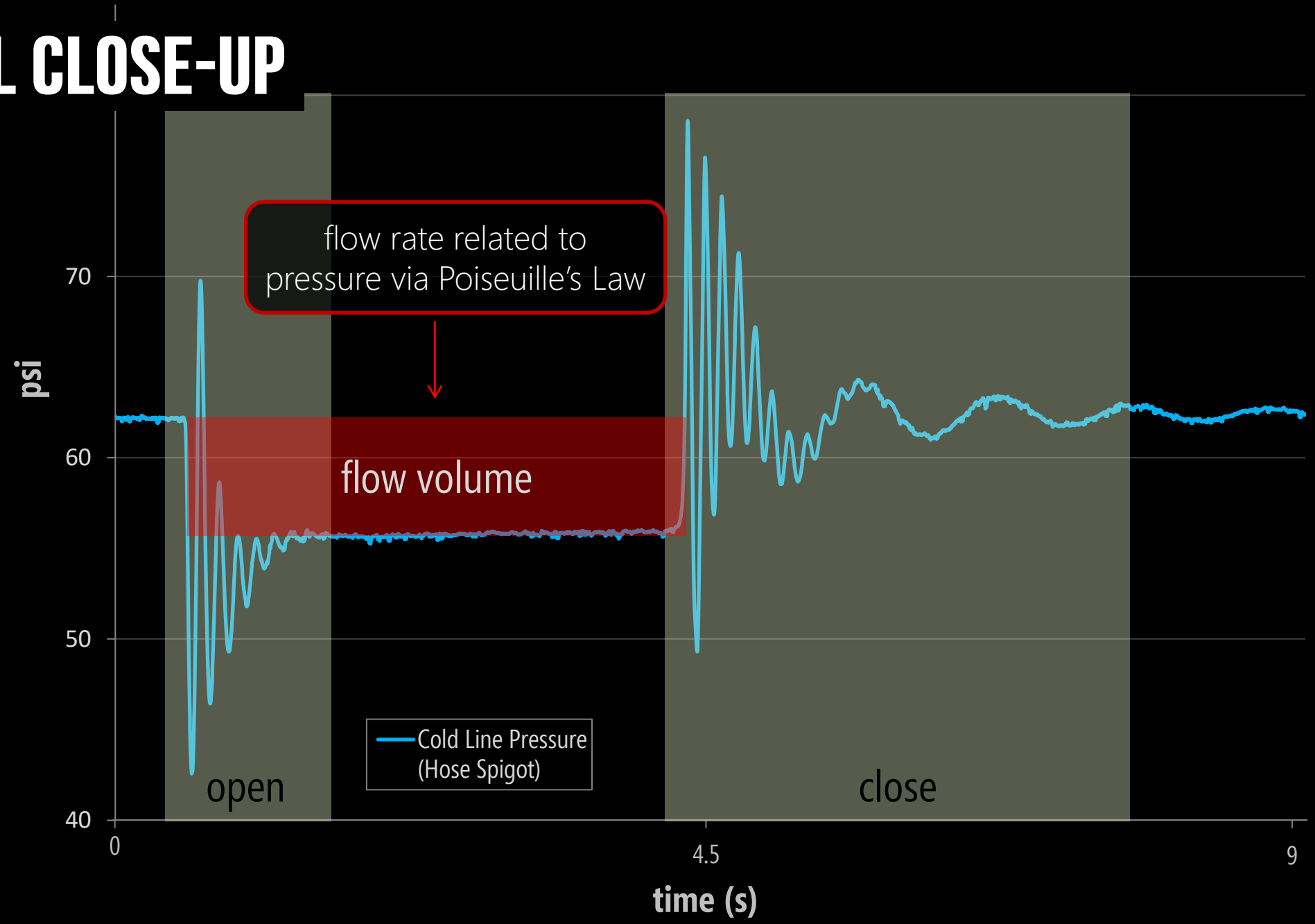

water tower
kitchen sink hot
bathroom 1
kitchen sink
hot open
kitchen
dishwasher
incoming cold
water from
supply line
utility water
meter
pressure
regulator
thermal
expansion
tank
hot
water
heater
laundry
bathroom 2

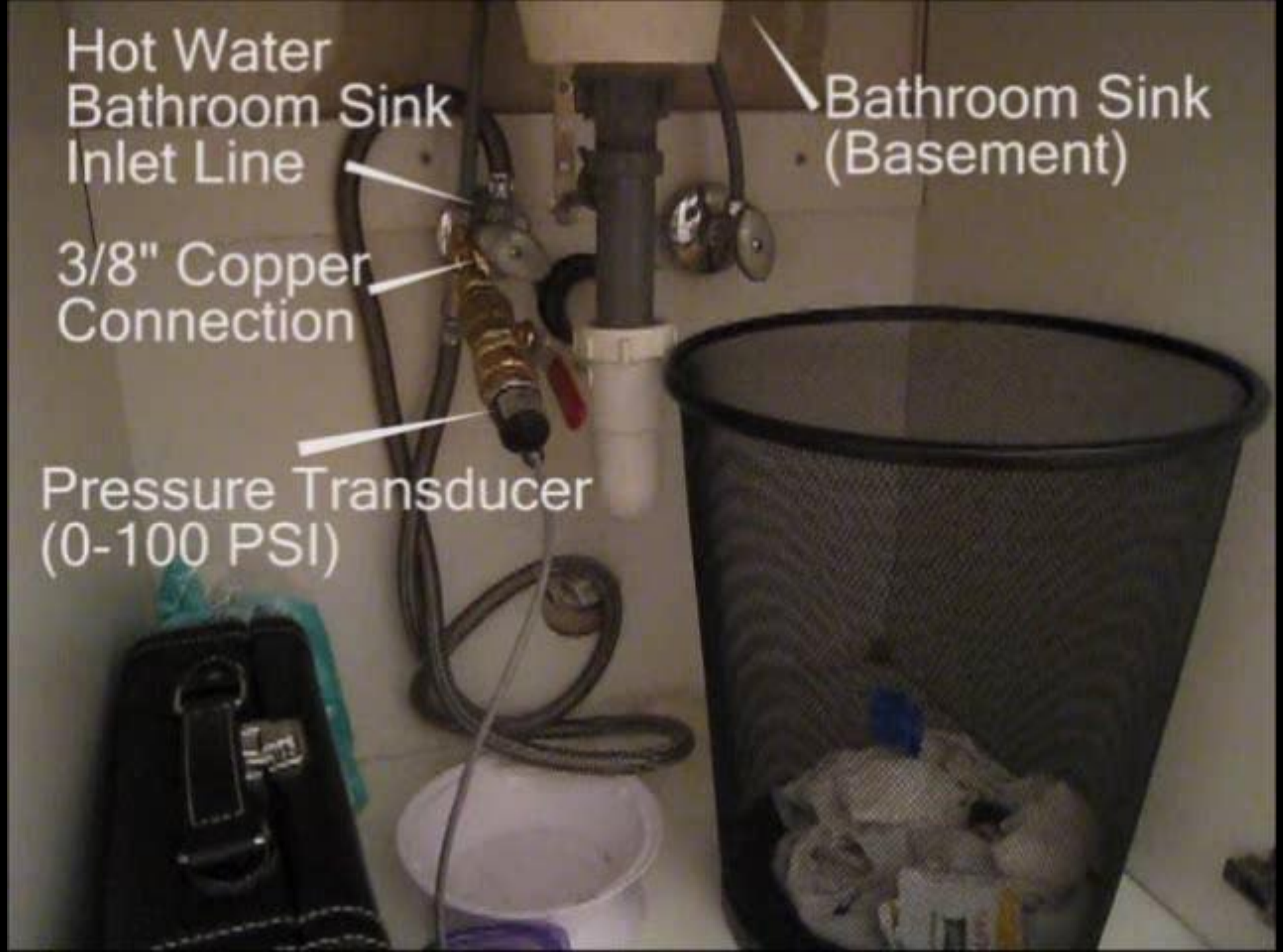
HYDROSENSE

SIGNAL CLOSE-UP

HYDROSENSE

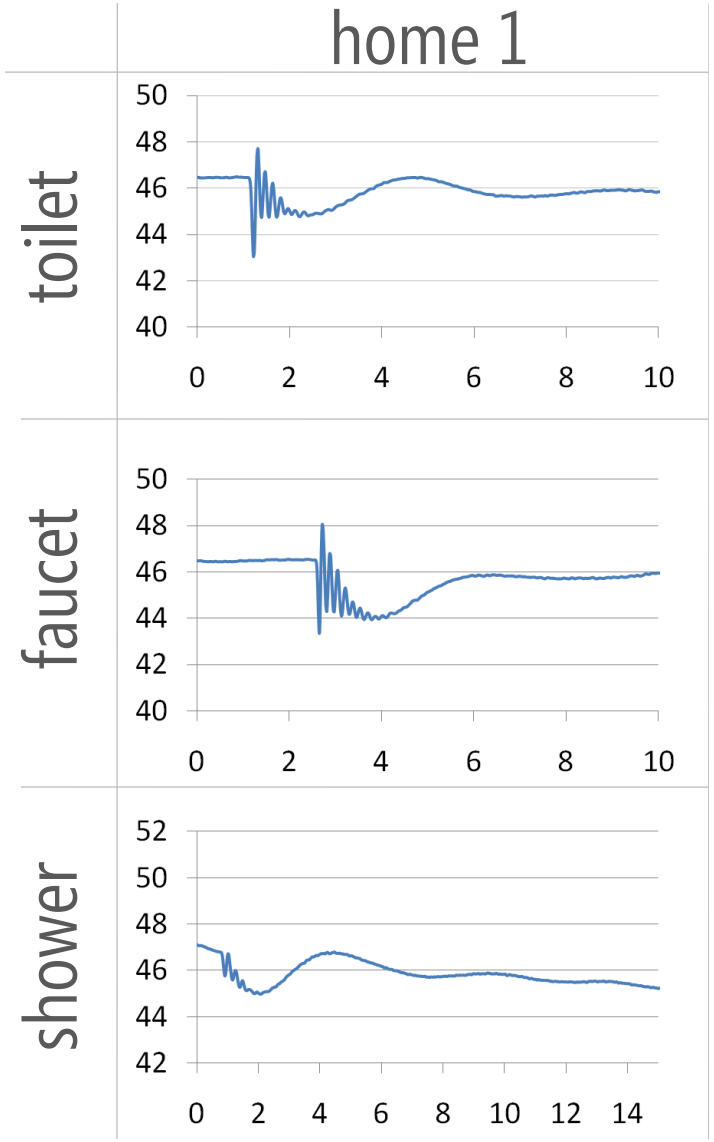
SIGNAL CLOSE-UP




Hot Water
Bathroom Sink
Inlet Line
3/8" Copper
Connection
Pressure Transducer
(0-100 PSI)
Bathroom Sink
(Basement)

HYDROSENSE

EXAMPLE OPEN EVENTS

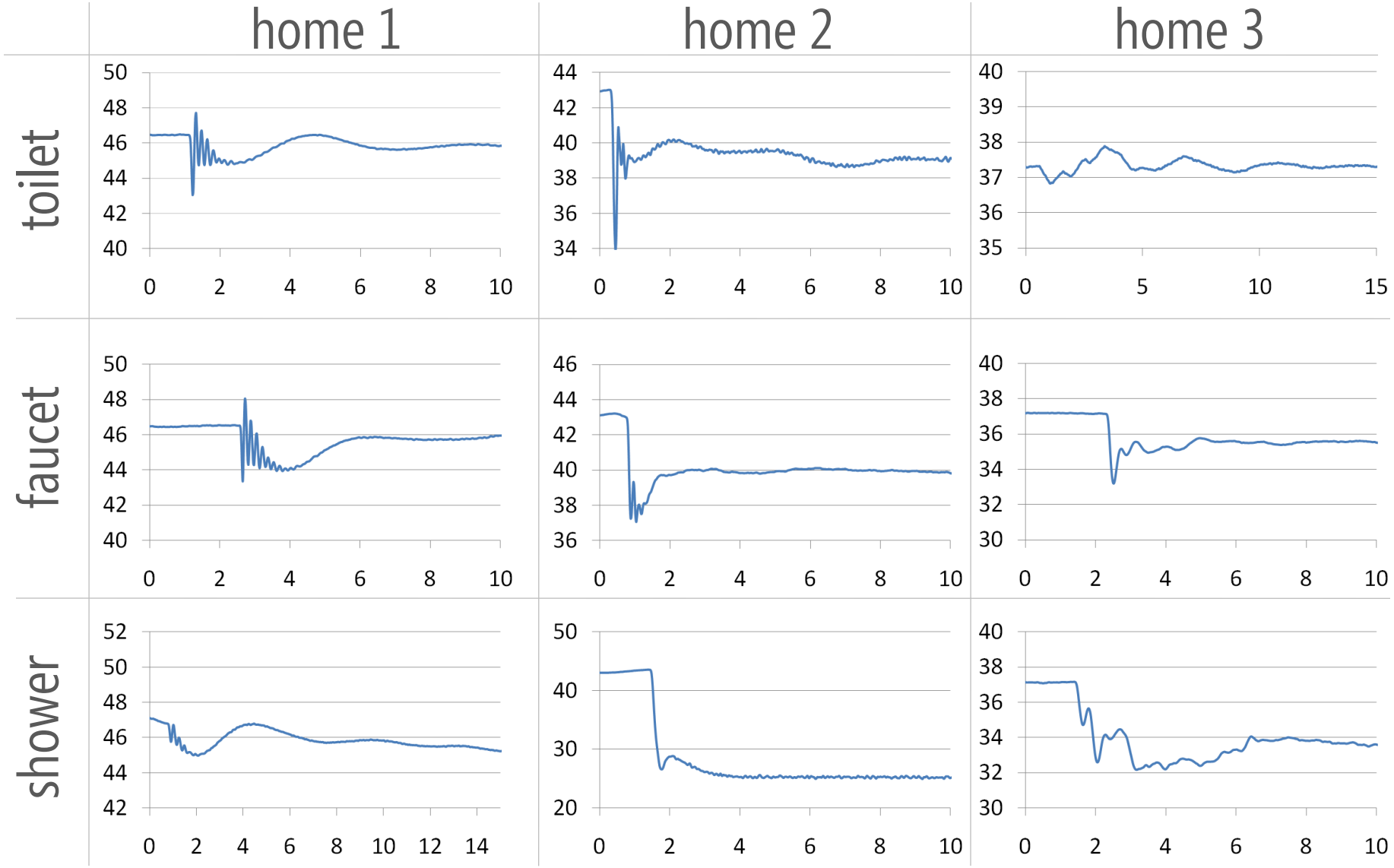


signature dependent on:

- fixture type
- valve type
- valve location in home

HYDROSENSE

EXAMPLE OPEN EVENTS



HYDROSENSE

CLASSIFICATION APPROACH

1. Install sensor
2. Collect example signals per valve (create a template library)
3. Monitor incoming signal
4. Detect that a water event has occurred (event detection + segmentation)
5. Classify event as ‘open’ or ‘close’
6. Classify source of event (*e.g.*, toilet, shower)
7. Estimate water flow volume

HYDROSENSE

CLASSIFICATION APPROACH

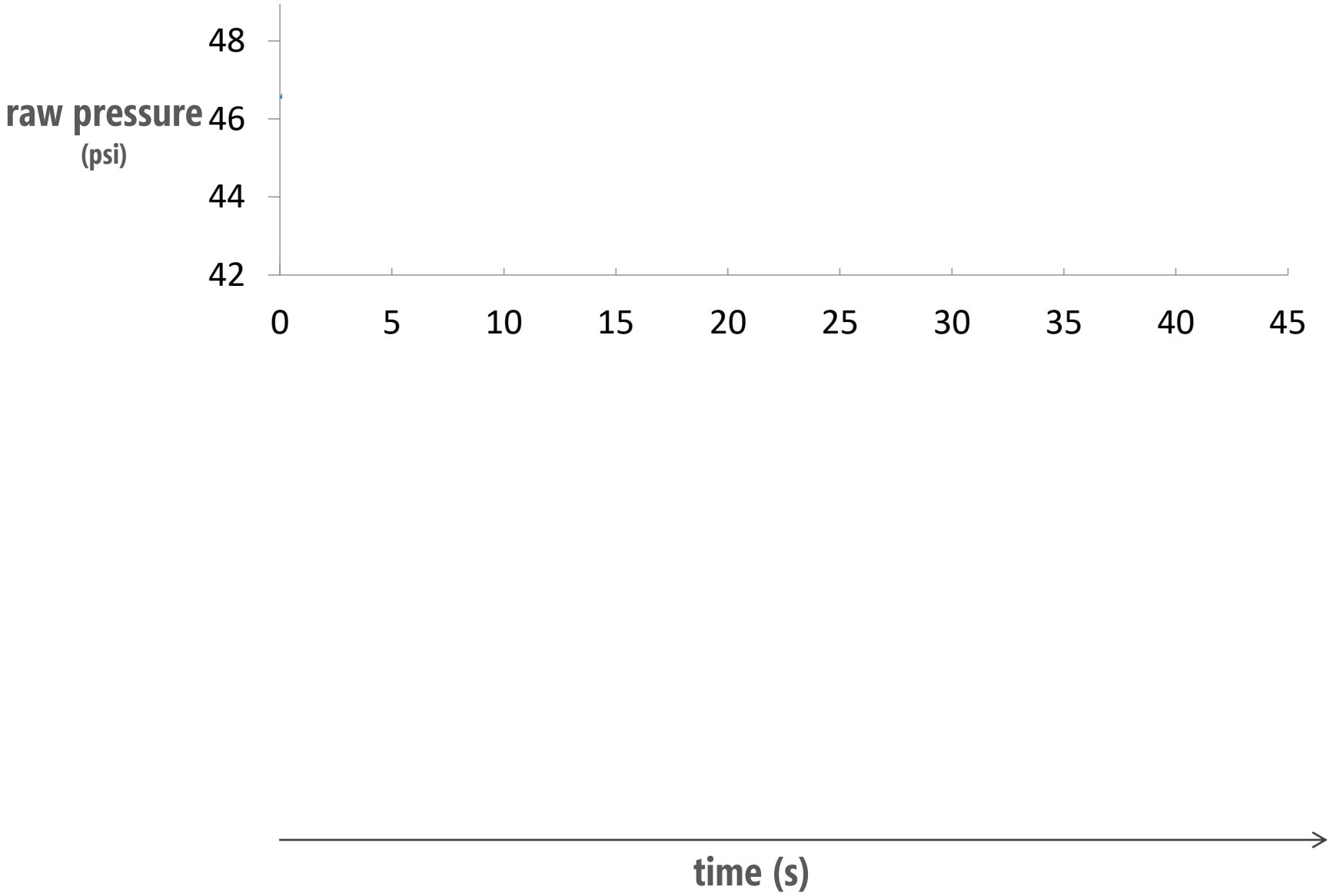
1. Install sensor
2. Collect example signals per valve (create a template library)
3. Monitor incoming signal
4. **Detect that a water event has occurred (event detection + segmentation)**
5. **Classify event as ‘open’ or ‘close’**
6. Classify source of event (*e.g.*, toilet, shower)
7. Estimate water flow volume

HYDROSENSE

EVENT DETECTION AND SEGMENTATION

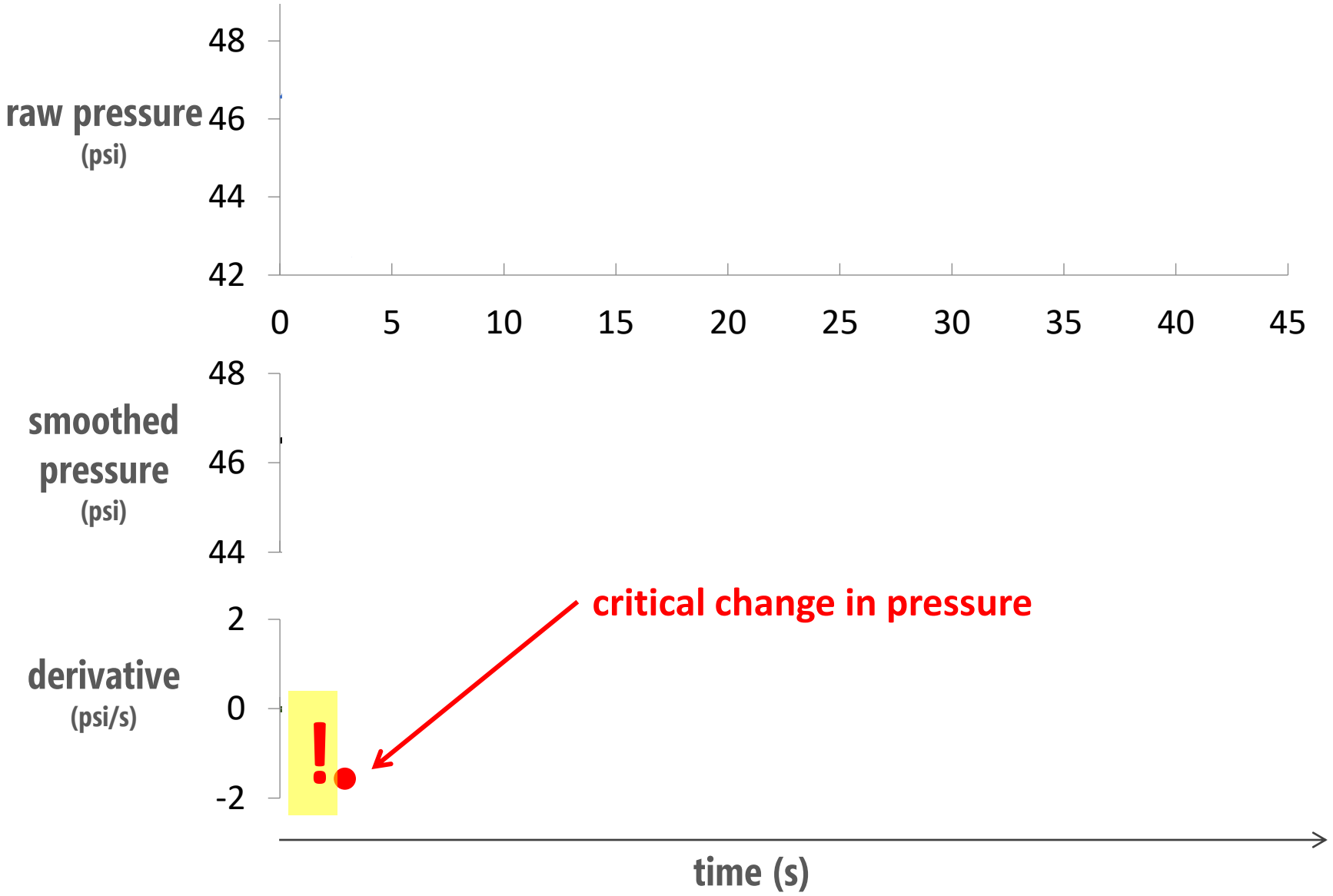
HYDROSENSE

EVENT DETECTION AND SEGMENTATION



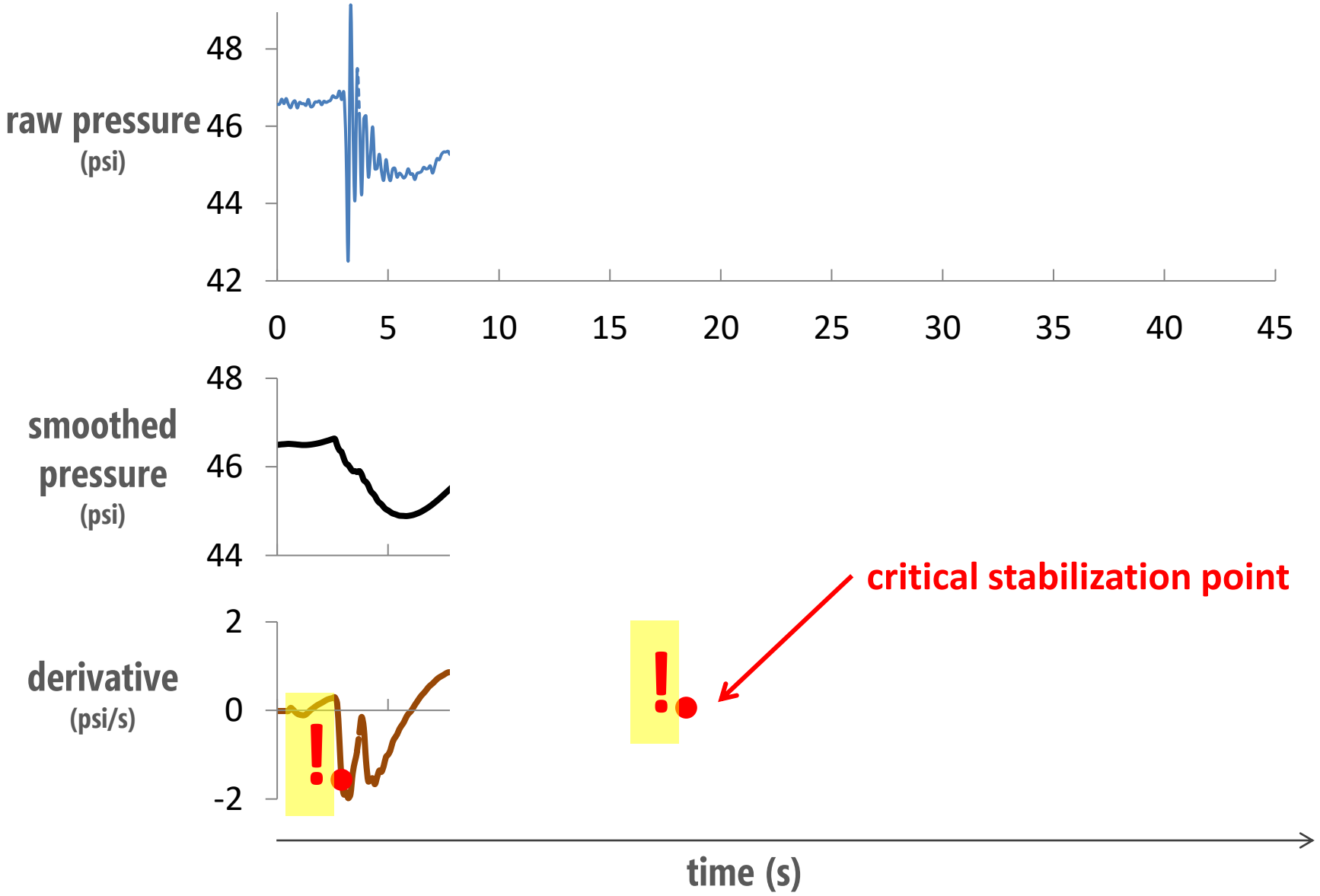
HYDROSENSE

EVENT DETECTION AND SEGMENTATION



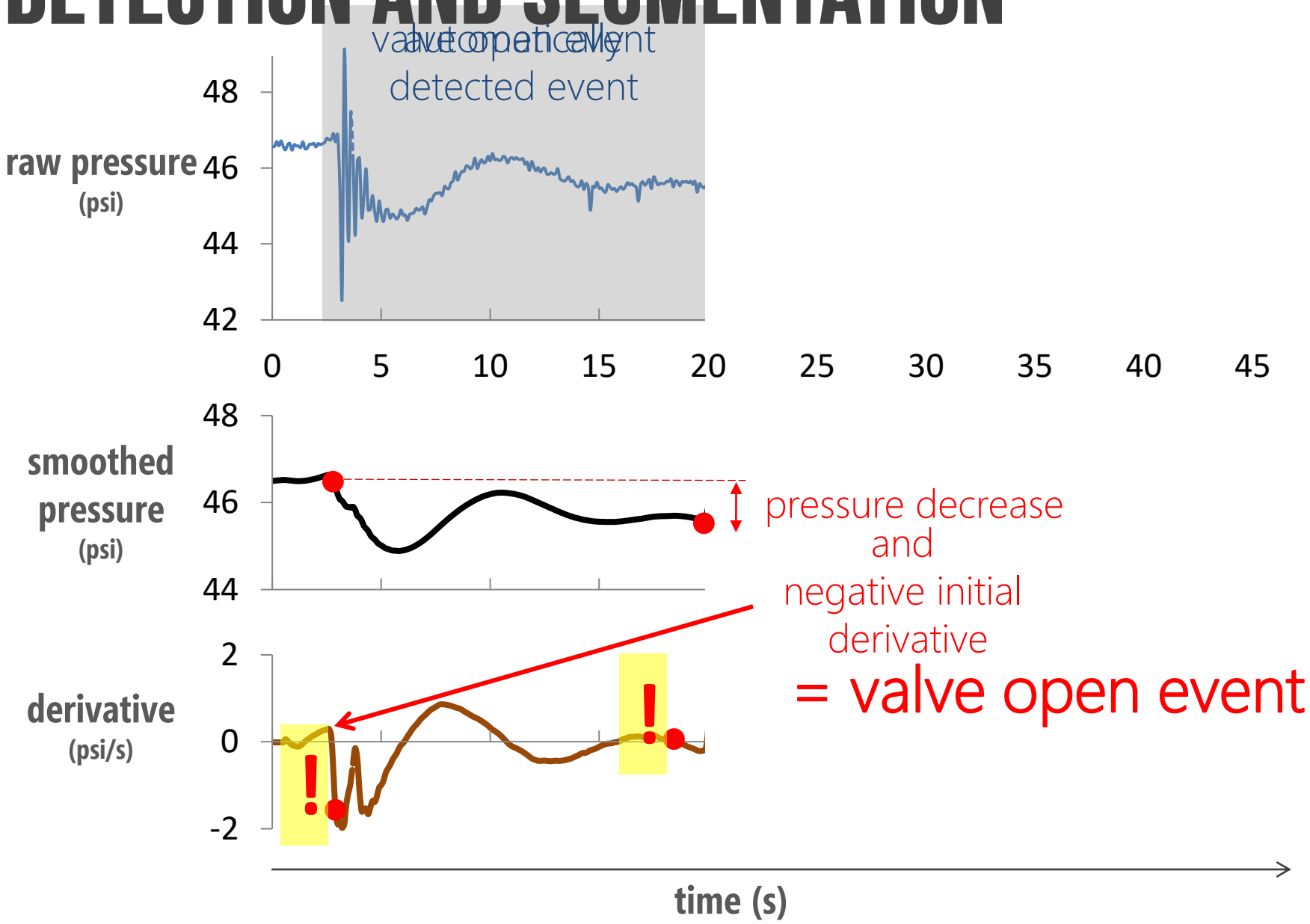
HYDROSENSE

EVENT DETECTION AND SEGMENTATION



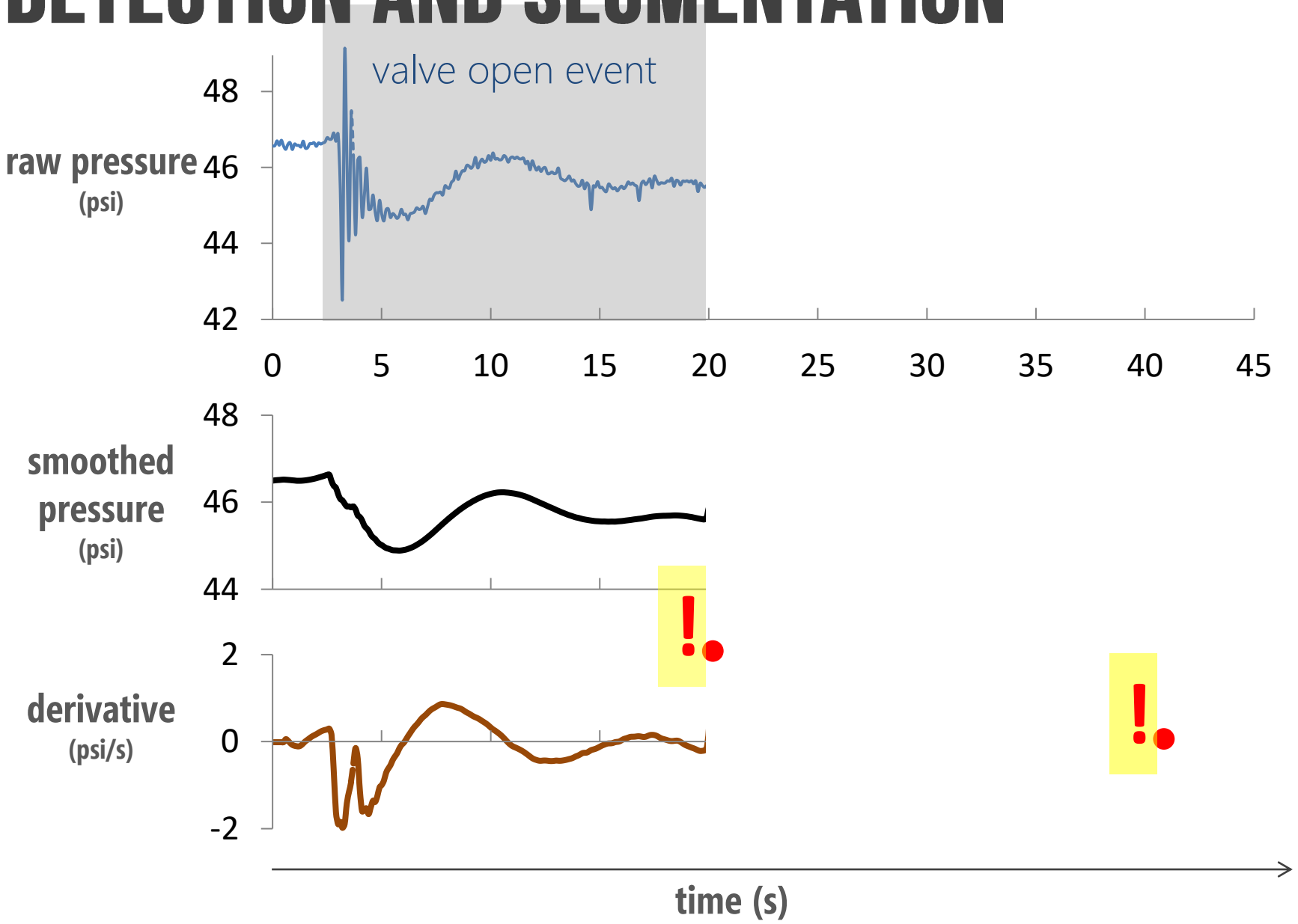
HYDROSENSE

EVENT DETECTION AND SEGMENTATION



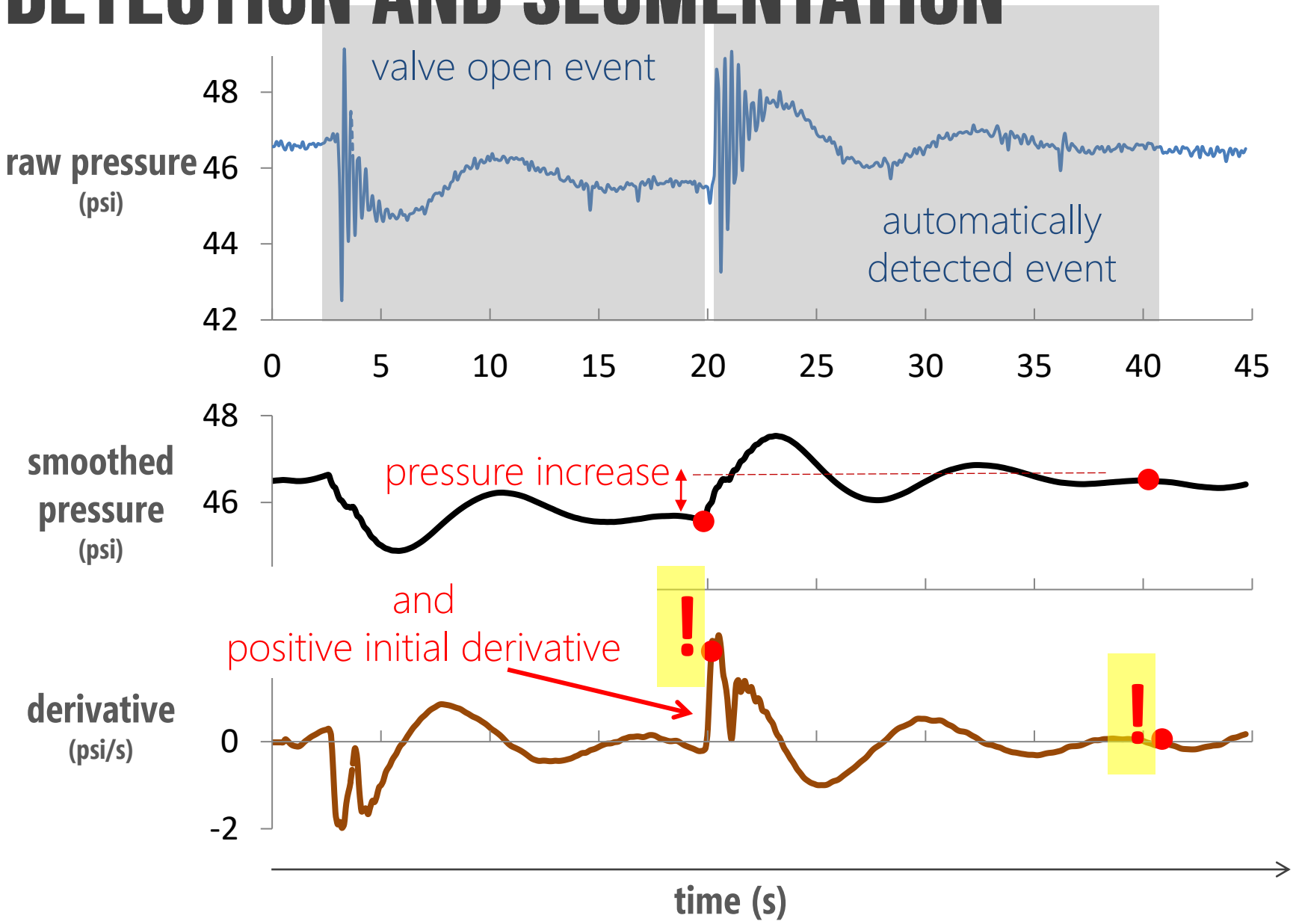
HYDROSENSE

EVENT DETECTION AND SEGMENTATION



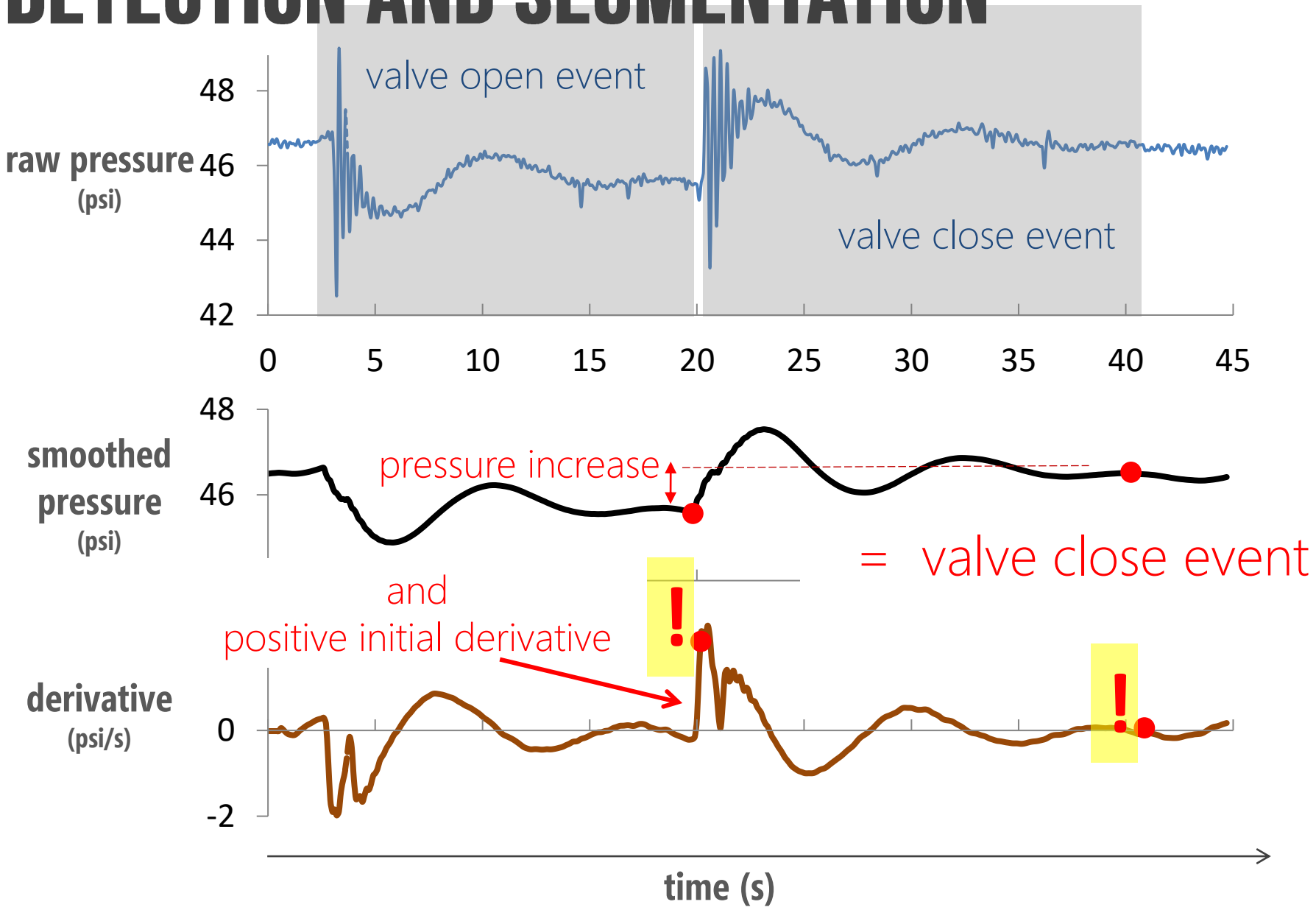
HYDROSENSE

EVENT DETECTION AND SEGMENTATION



HYDROSENSE

EVENT DETECTION AND SEGMENTATION



HYDROSENSE

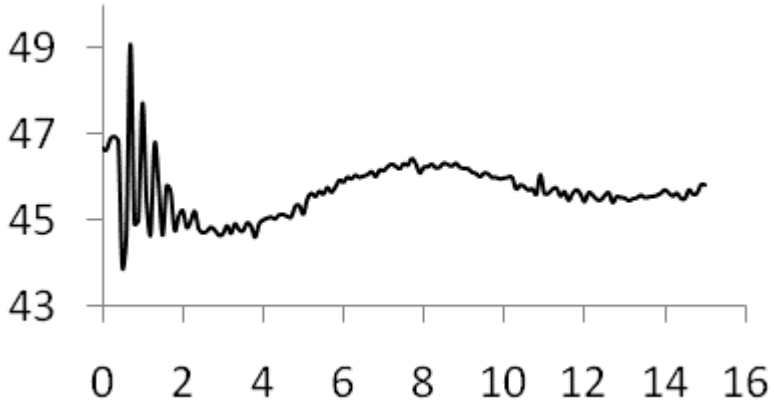
CLASSIFICATION APPROACH

1. Install sensor
2. Collect example signals per valve (create a template library)
3. Monitor incoming signal
4. Detect that a water event has occurred (event detection + segmentation)
5. Classify event as ‘open’ or ‘close’
6. **Classify source of event (*e.g.,* toilet, shower)**
7. Estimate water flow volume

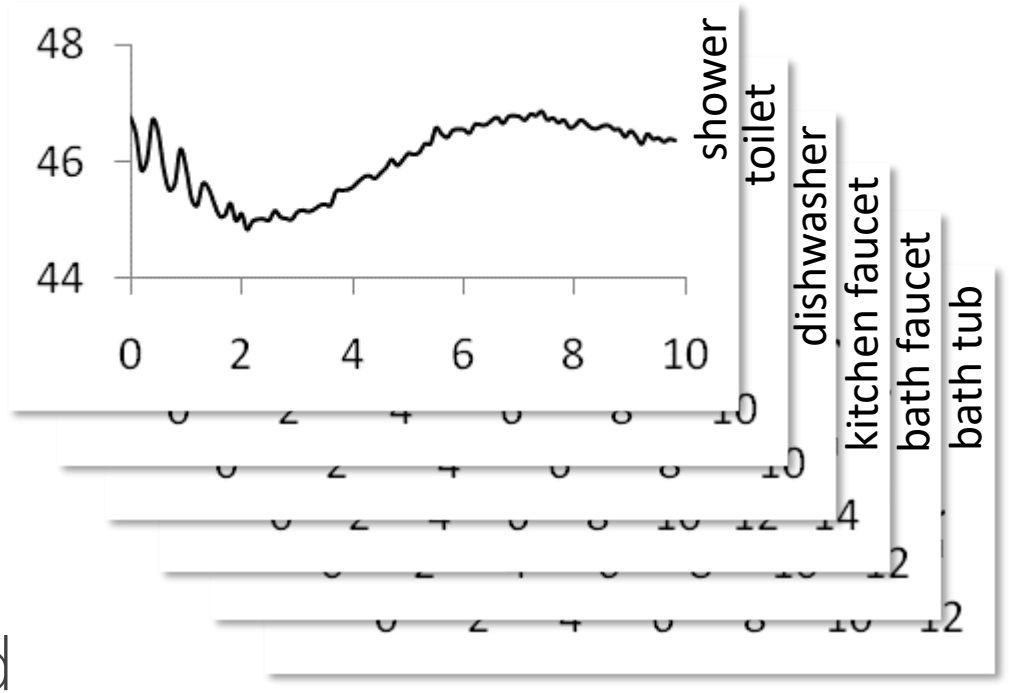
HYDROSENSE

CLASSIFY FIXTURES VIA TEMPLATE MATCHING

Incoming Signal
Segmented open event



Templates
Library of labeled open events



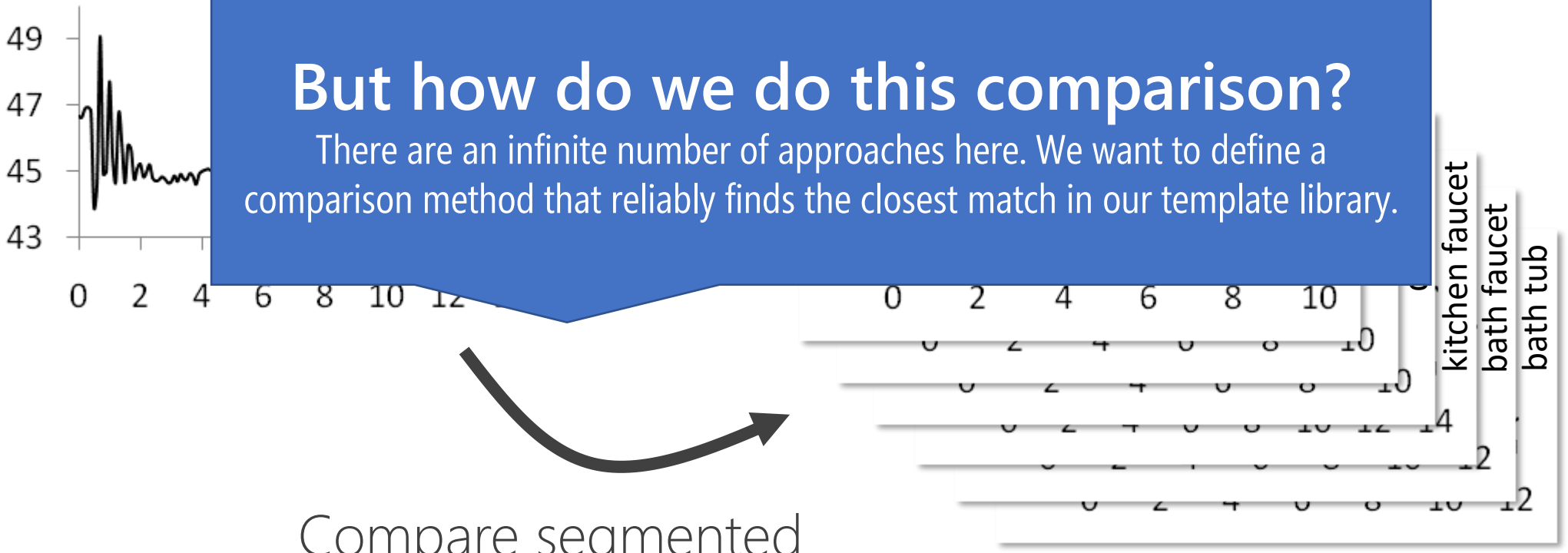
Compare segmented event to stored archive of labeled templates

HYDROSENSE

CLASSIFY FIXTURES VIA TEMPLATE MATCHING

Incoming Signal
Segmented open event

Templates
Library of labeled open events



But how do we do this comparison?

There are an infinite number of approaches here. We want to define a comparison method that reliably finds the closest match in our template library.

Compare segmented event to stored archive of labeled templates

HYDROSENSE

MATCHED FILTERING

matched fil·ter

In signal processing, a matched filter is obtained by correlating a known signal (or template) with an unknown signal to detect the presence of the template in the unknown signal.

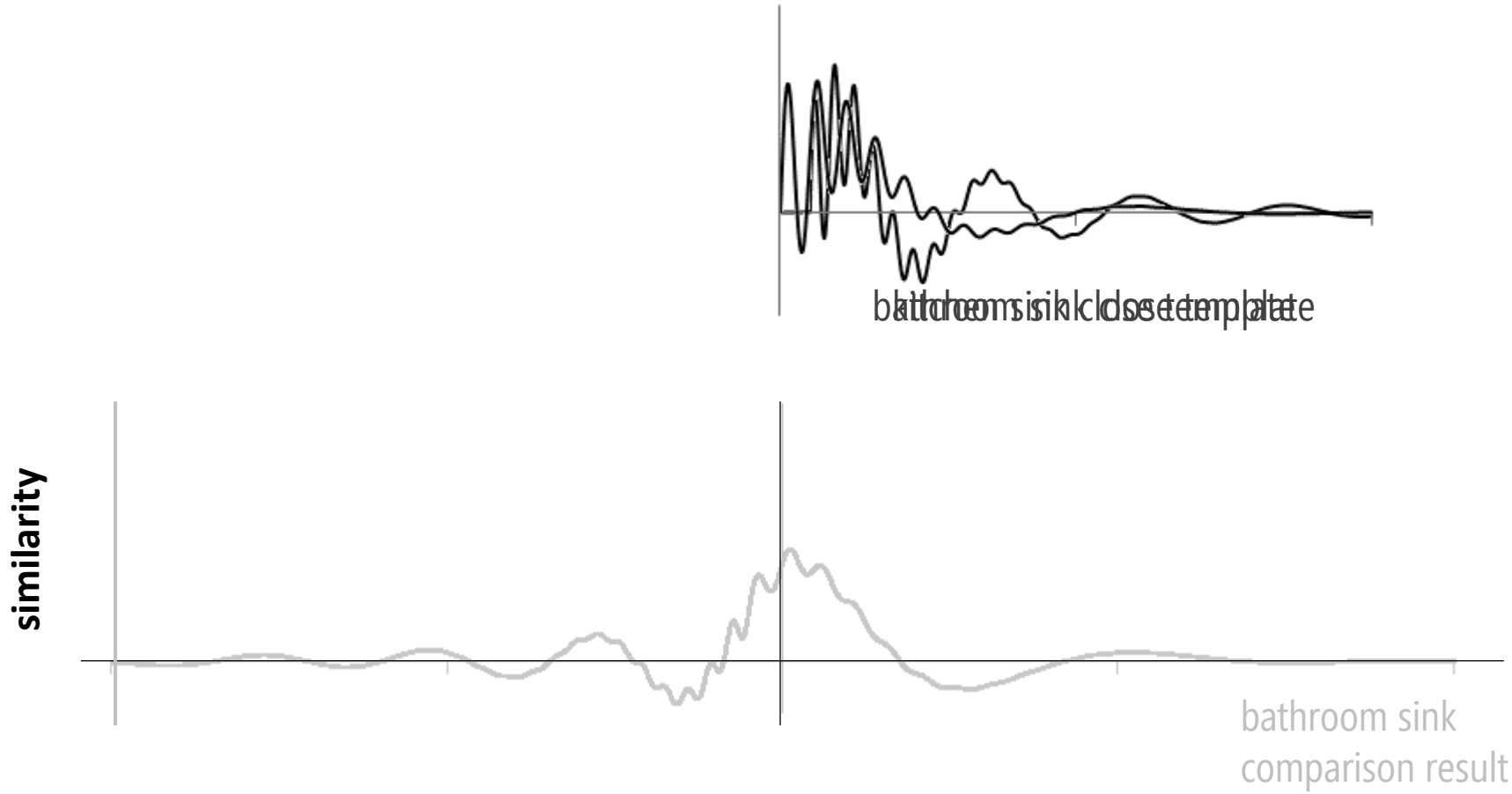
- Wikipedia

HYDROSENSE

MATCHED FILTERING

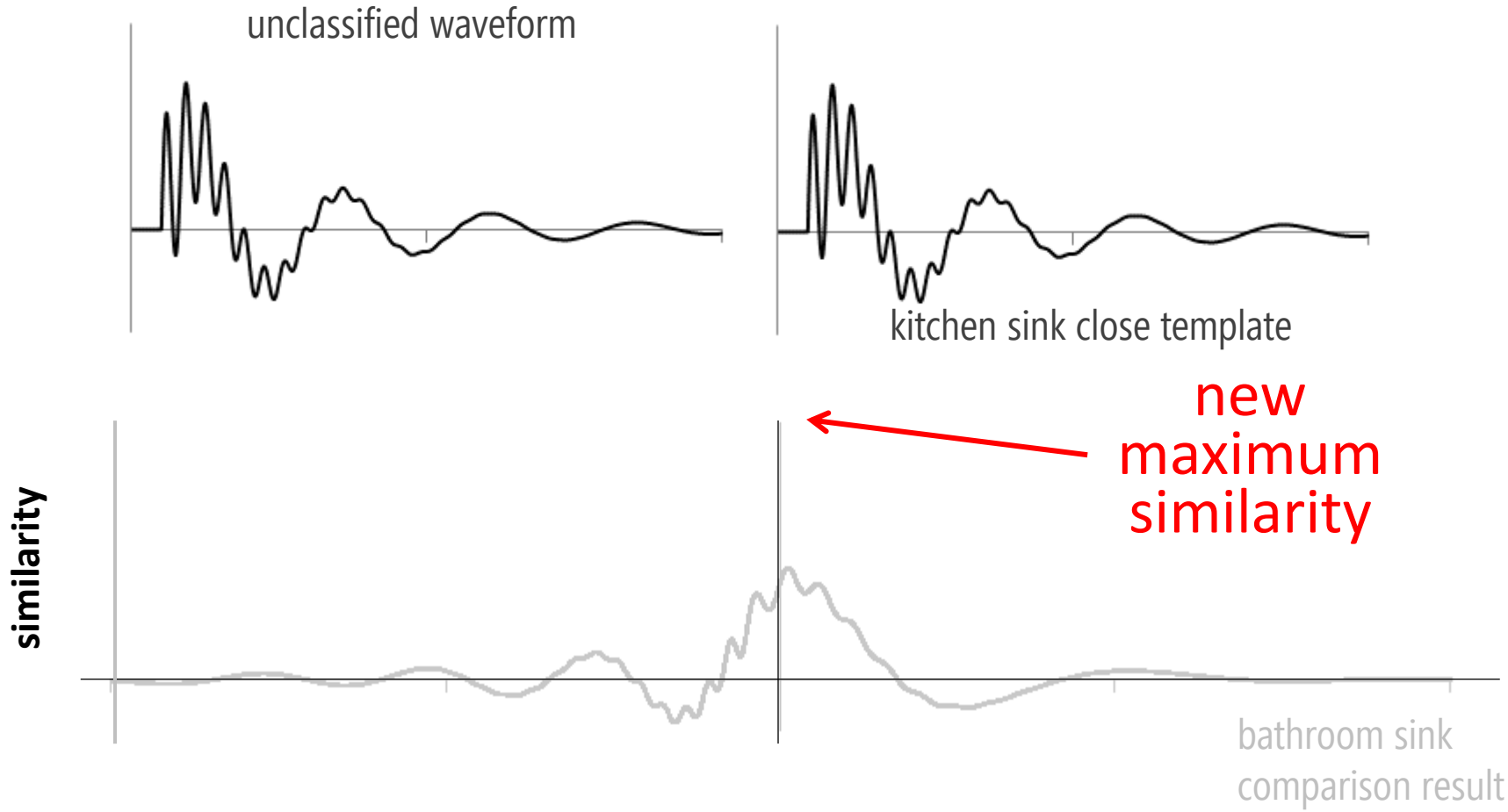
HYDROSENSE

MATCHED FILTERING



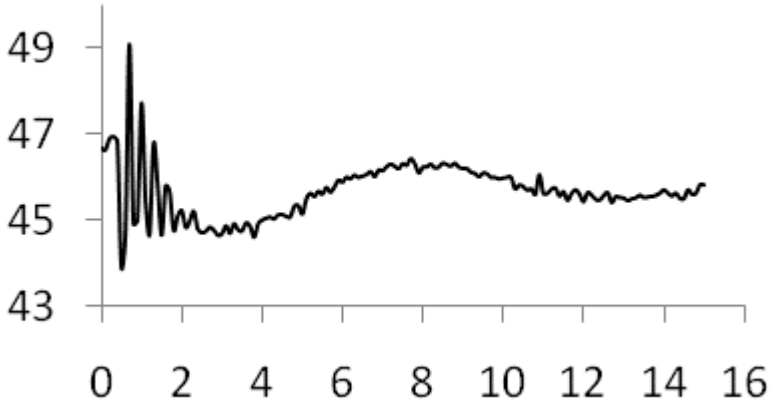
HYDROSENSE

MATCHED FILTERING

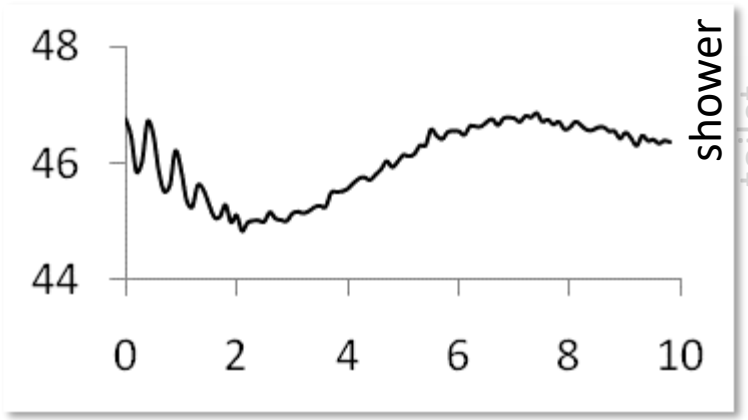


But we didn’t compare the raw signals. We compared three signal transformations.

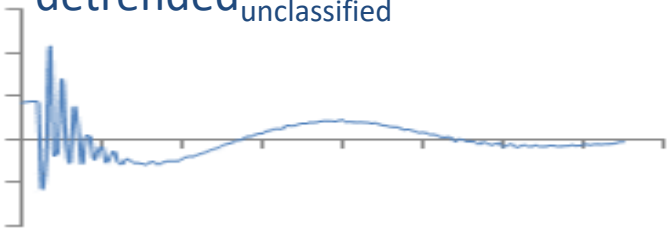
unclassified open event



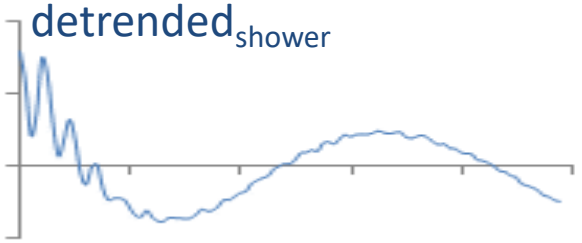
open event library



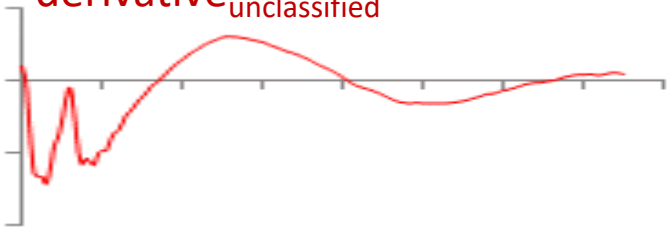
detrended$_{unclassified}$



detrended$_{shower}$

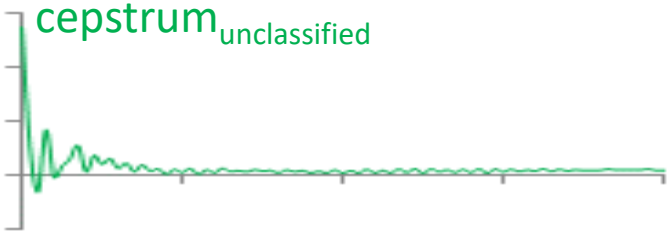


derivative$_{unclassified}$

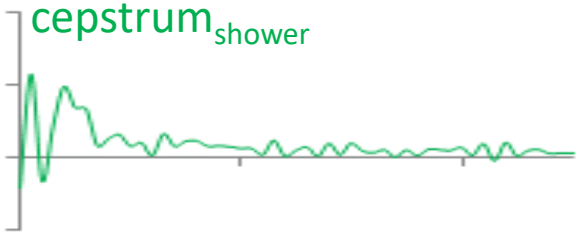


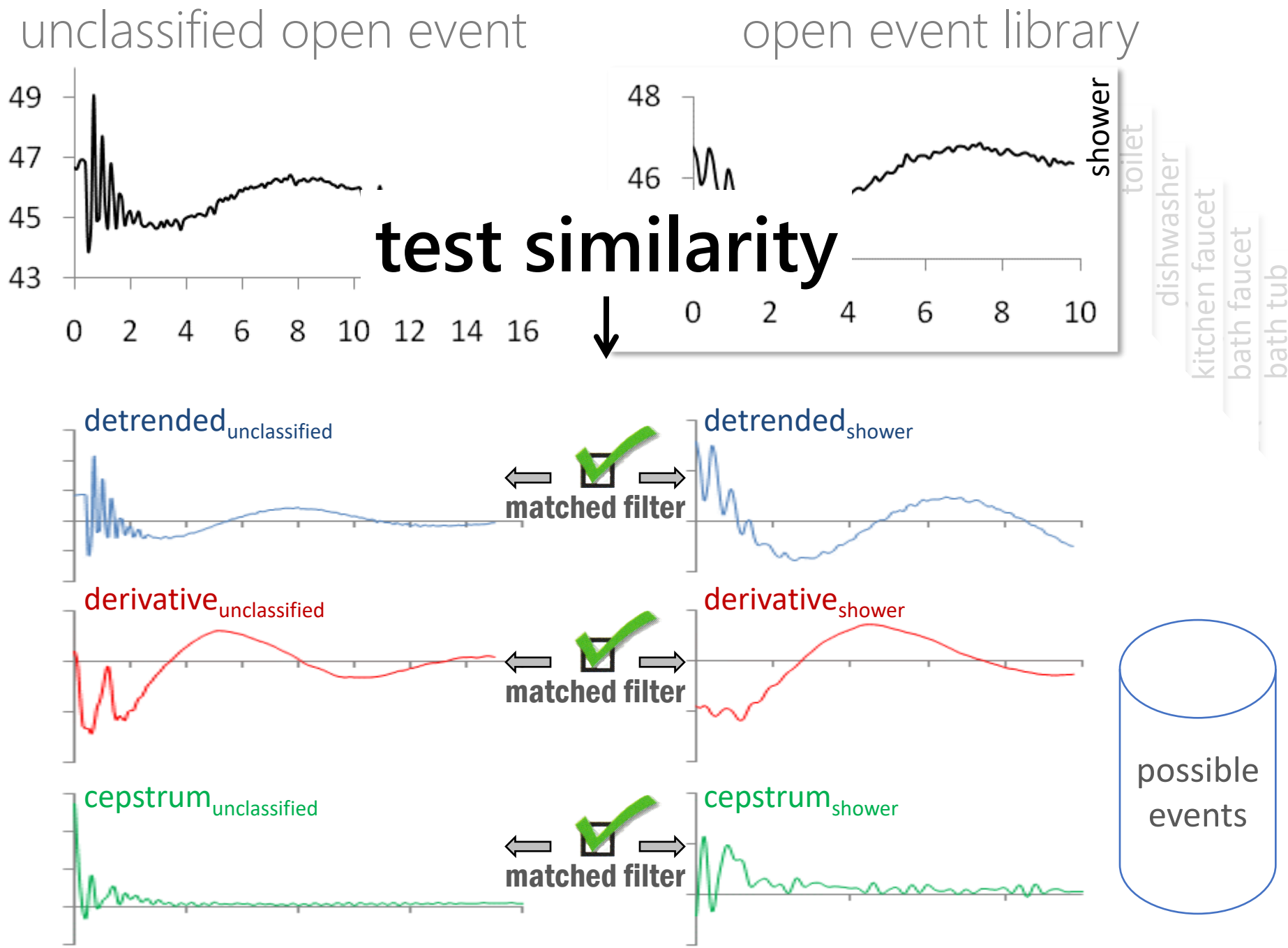
derivative$_{shower}$

cepstrum$_{unclassified}$

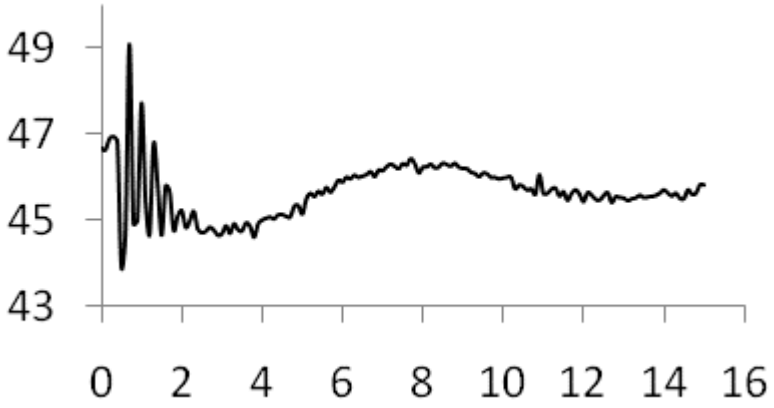


cepstrum$_{shower}$

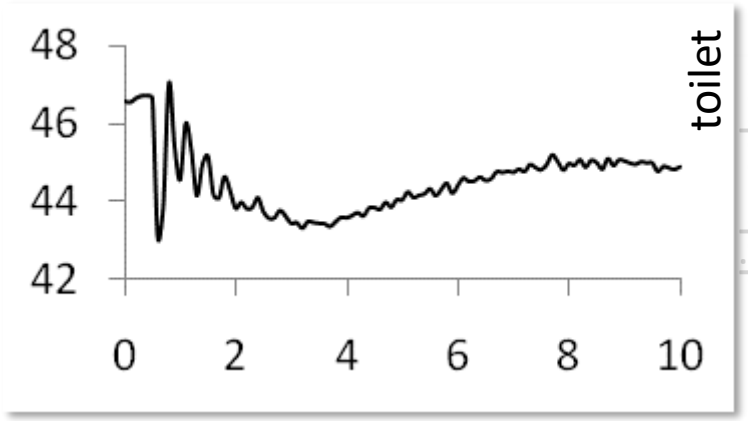


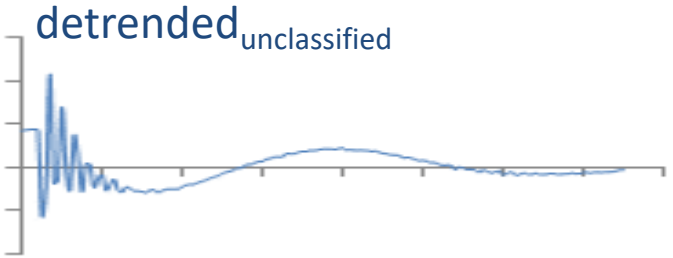

unclassified open event
open event library
49
47
45
43
0 2 4 6 8 10 12 14 16
48
46
shower
toilet
dishwasher
kitchen faucet
bath faucet
bath tub
0 2 4 6 8 10
test similarity
detrended unclassified
detrended shower
matched filter
derivative unclassified
derivative shower
matched filter
cepstrum unclassified
cepstrum shower
matched filter
possible events

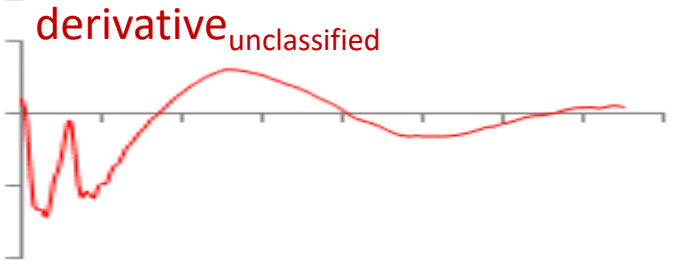
unclassified open event

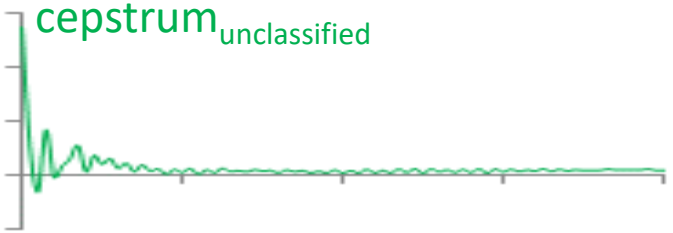


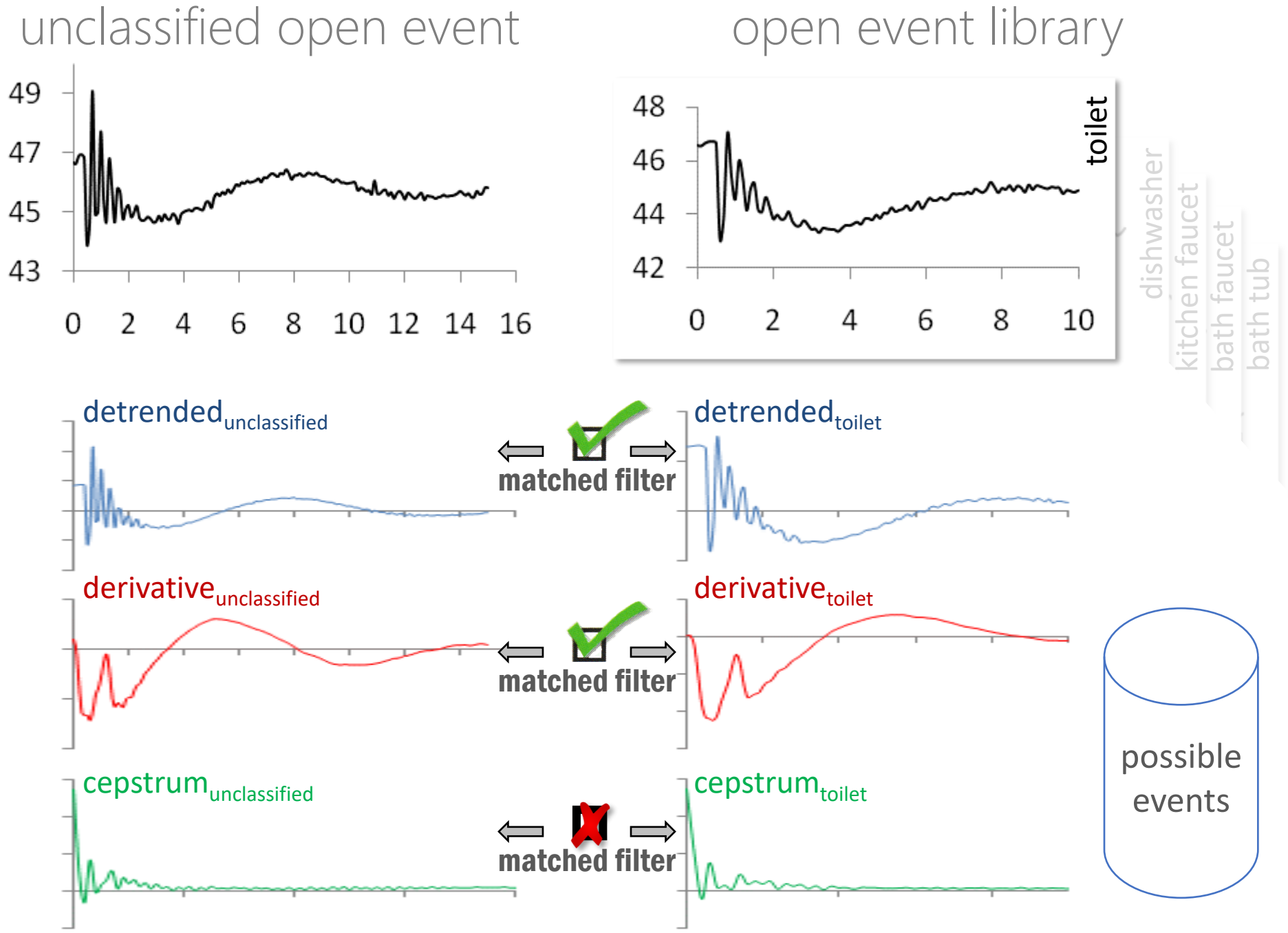
open event library



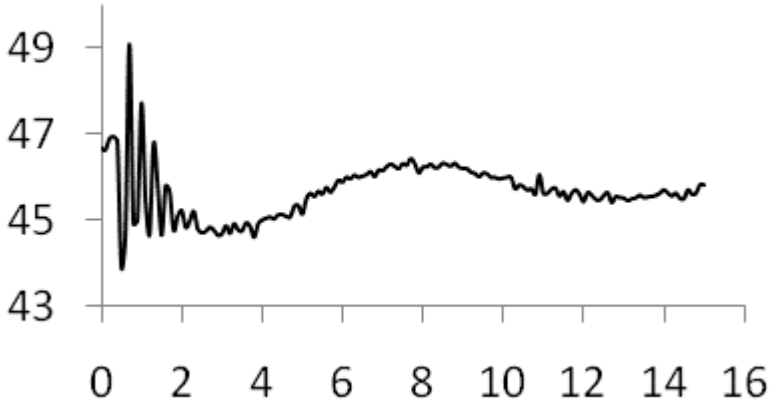




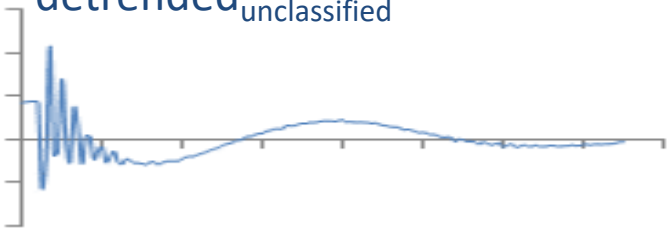



unclassified open event
open event library
49
47
45
43
0 2 4 6 8 10 12 14 16
48
46
44
42
0 2 4 6 8 10
toilet
dishwasher
kitchen faucet
bath faucet
bath tub
detrended unclassified
matched filter
detrended toilet
derivative unclassified
matched filter
derivative toilet
cepstrum unclassified
matched filter
cepstrum toilet
possible events

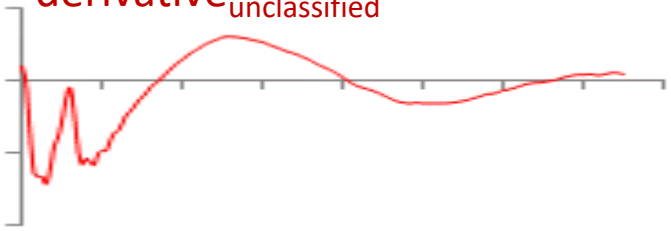
unclassified open event



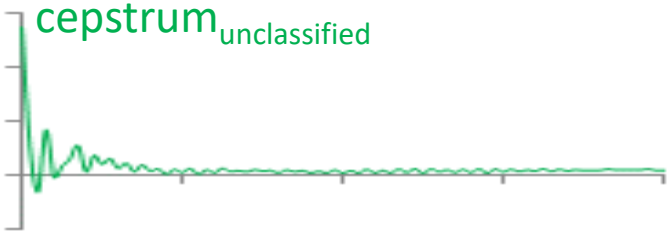
detrended$_{unclassified}$



derivative$_{unclassified}$



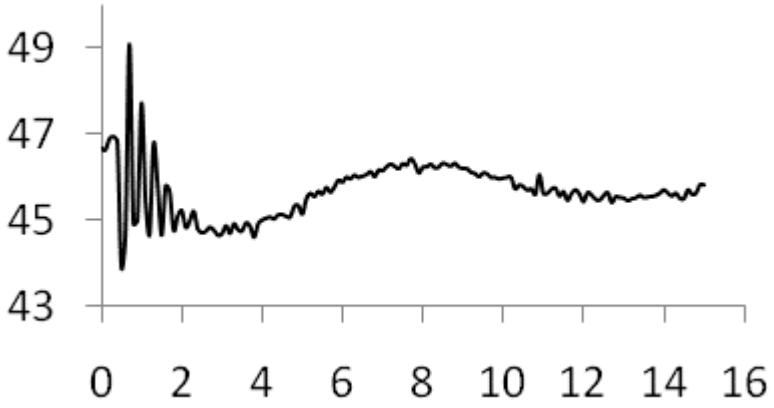
cepstrum$_{unclassified}$

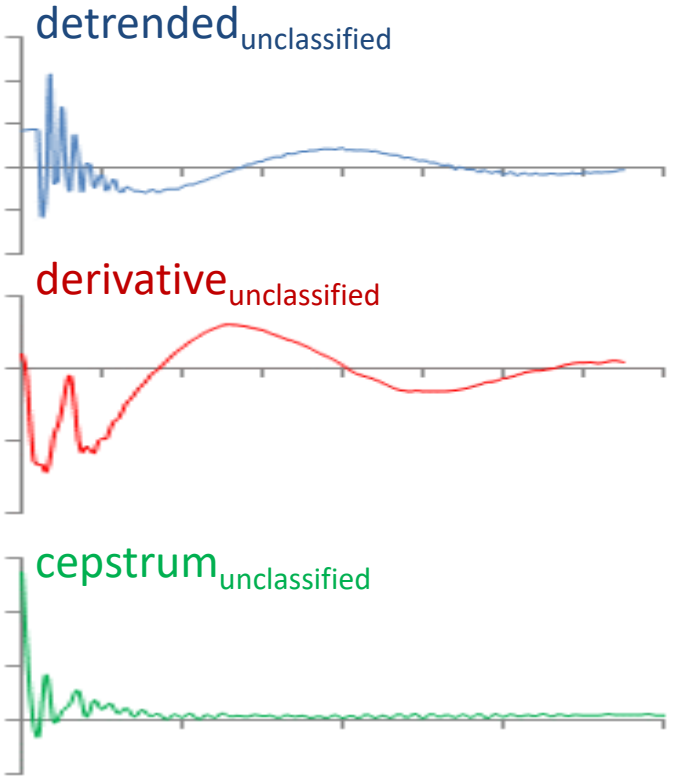


open event library

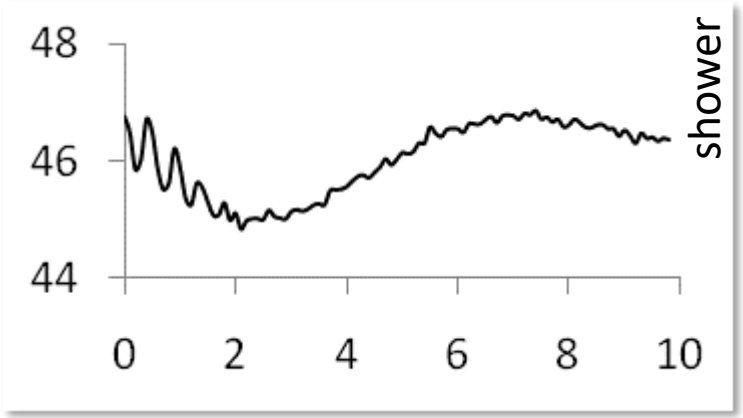
possible events

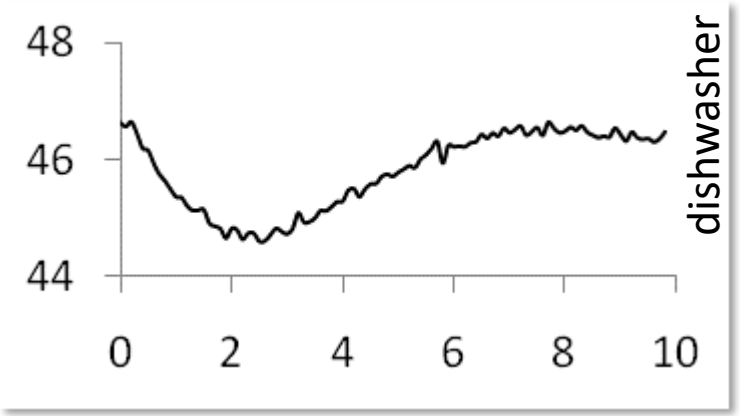
unclassified open event

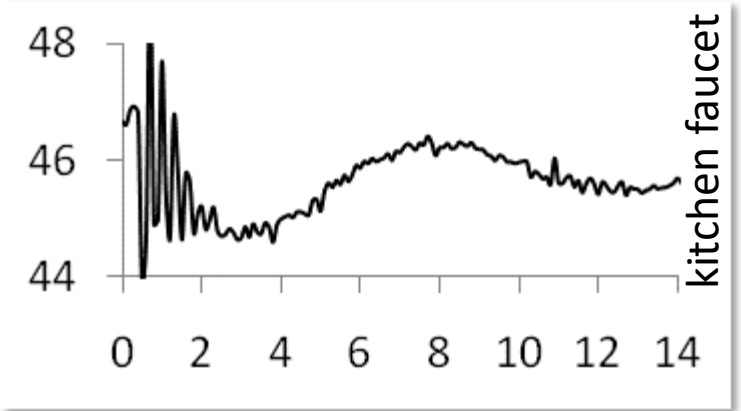




open event library

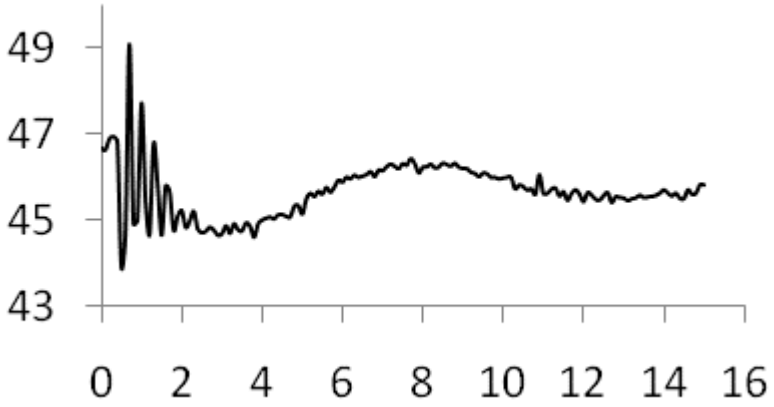


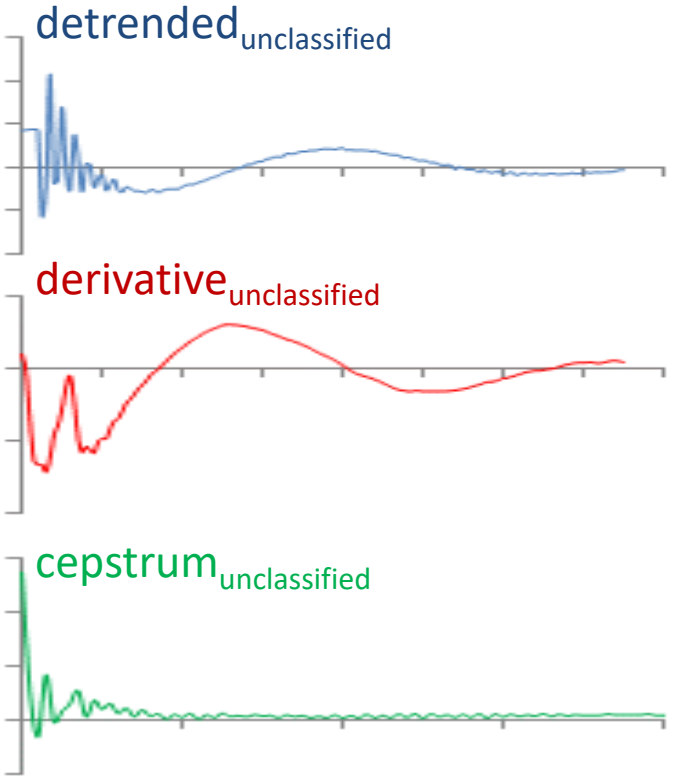




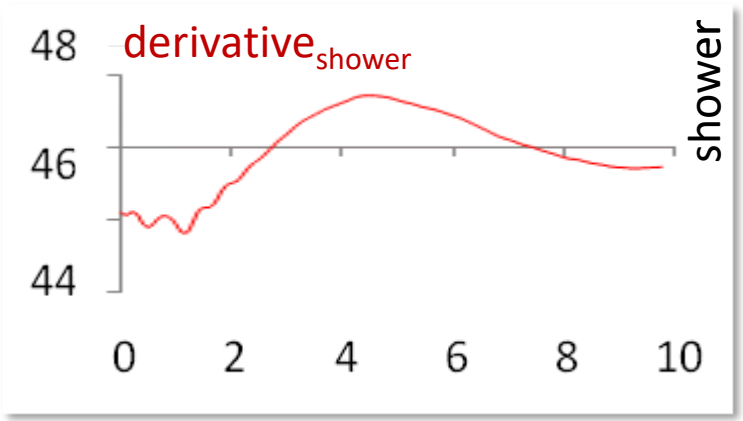
possible events

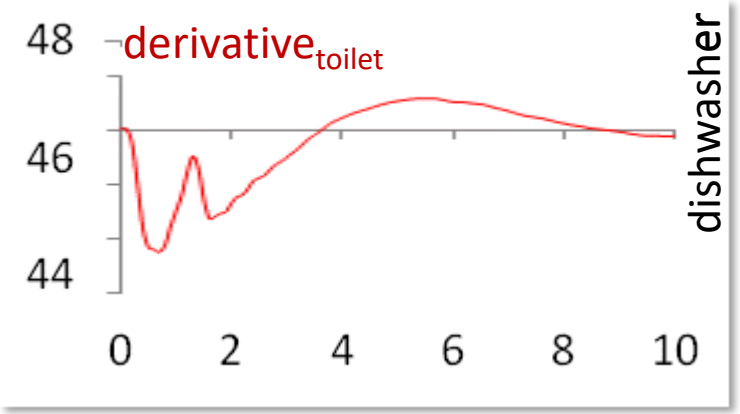
unclassified open event

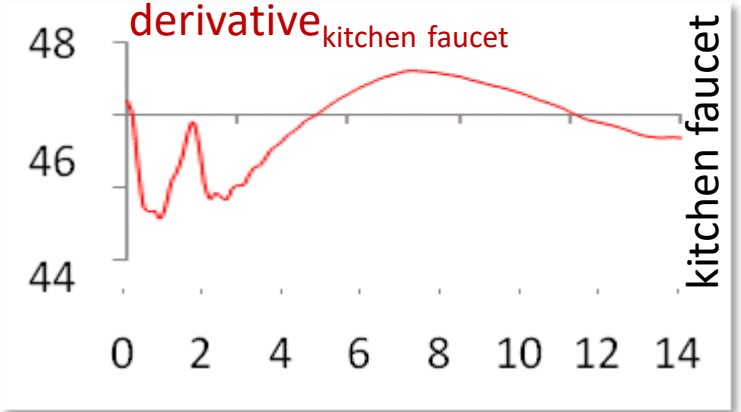




open event library

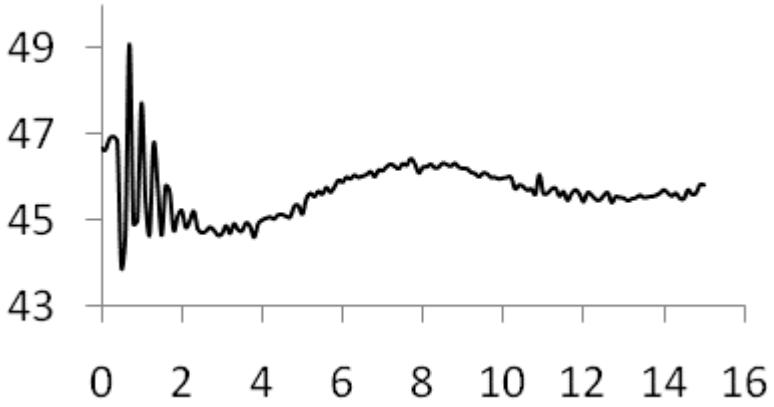




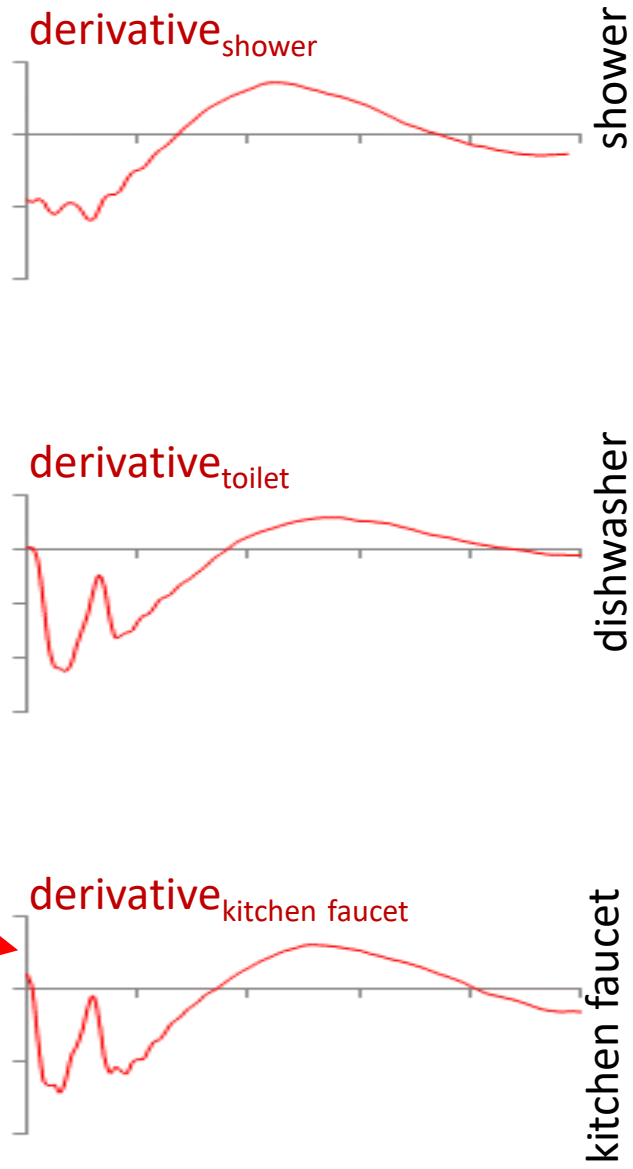


possible events

unclassified open event



open event library



Nearest neighbor match

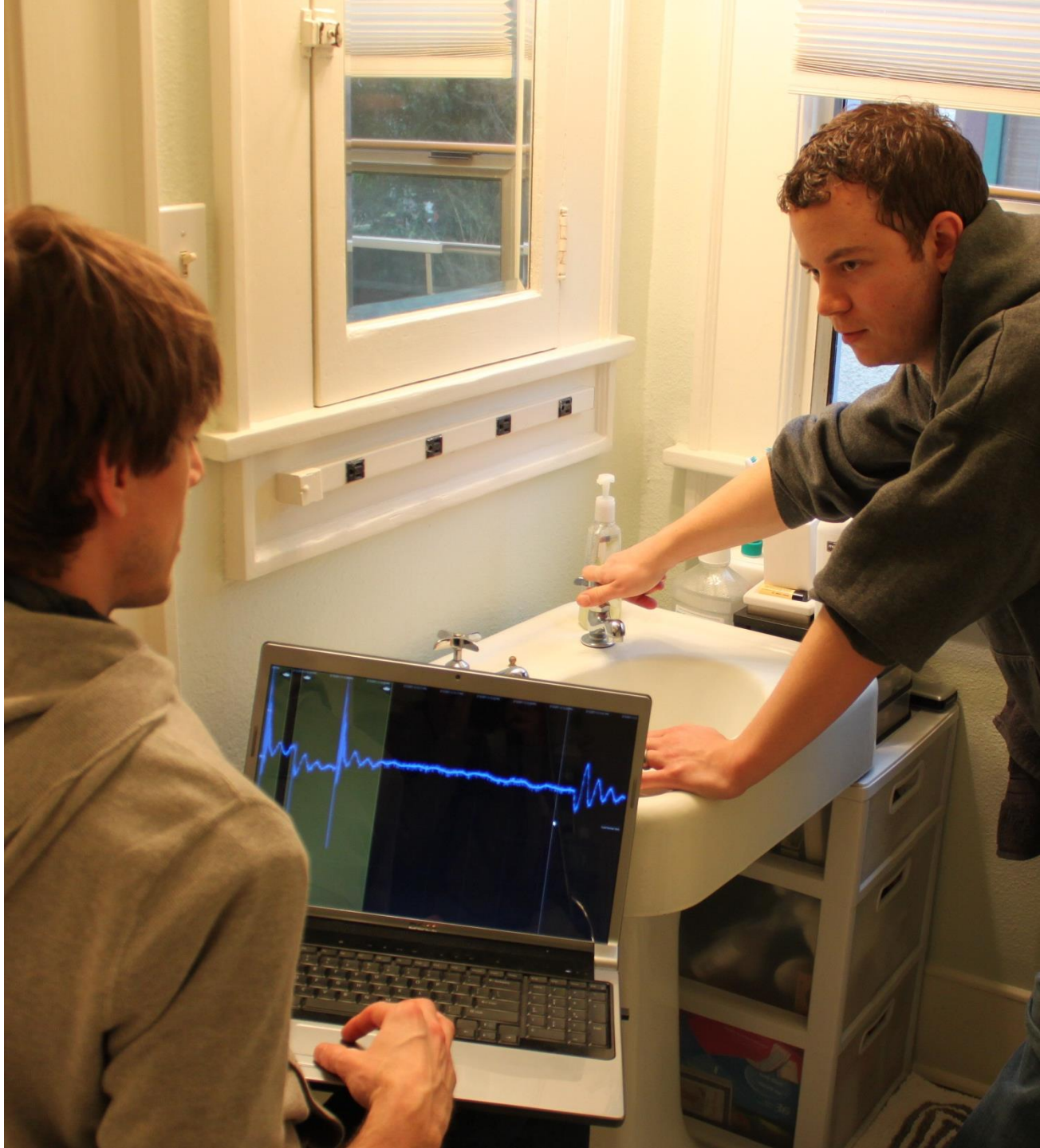
HYDROSENSE

EVALUATION

Visited 10 homes

Collected 5 trials per valve

For each trial, fully open valve for 5 seconds



HYDROSENSE

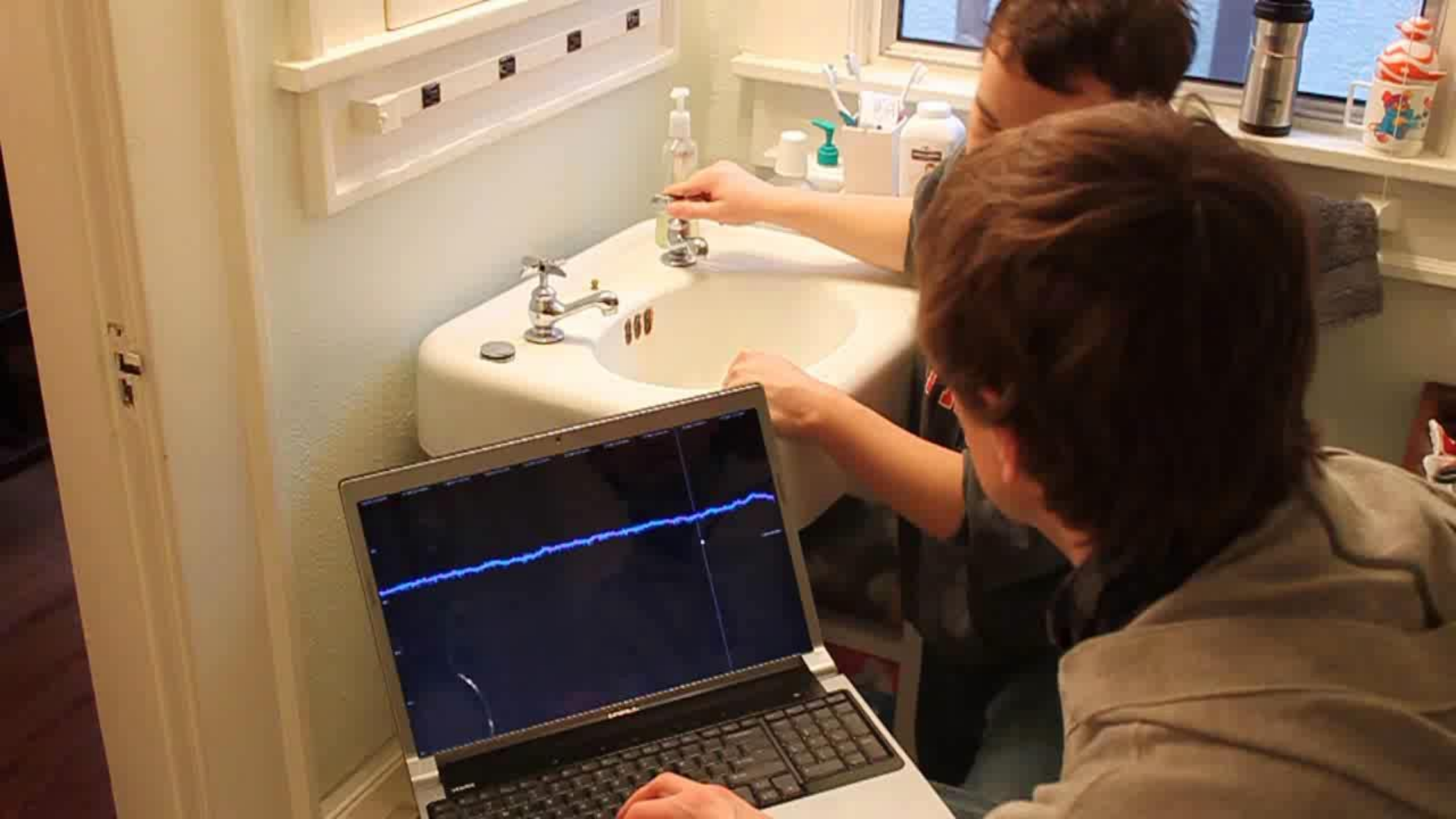
EVALUATION

Visited 10 homes

Collected 5 trials per valve

For each trial, fully open valve for 5 seconds

For 4 of the 10 homes, also gathered flow data: measure time to fill 1 gallon in a calibrated bucket



HYDROSENSE

DATA COLLECTION STATS

10 homes

706 trials

155 flow trials

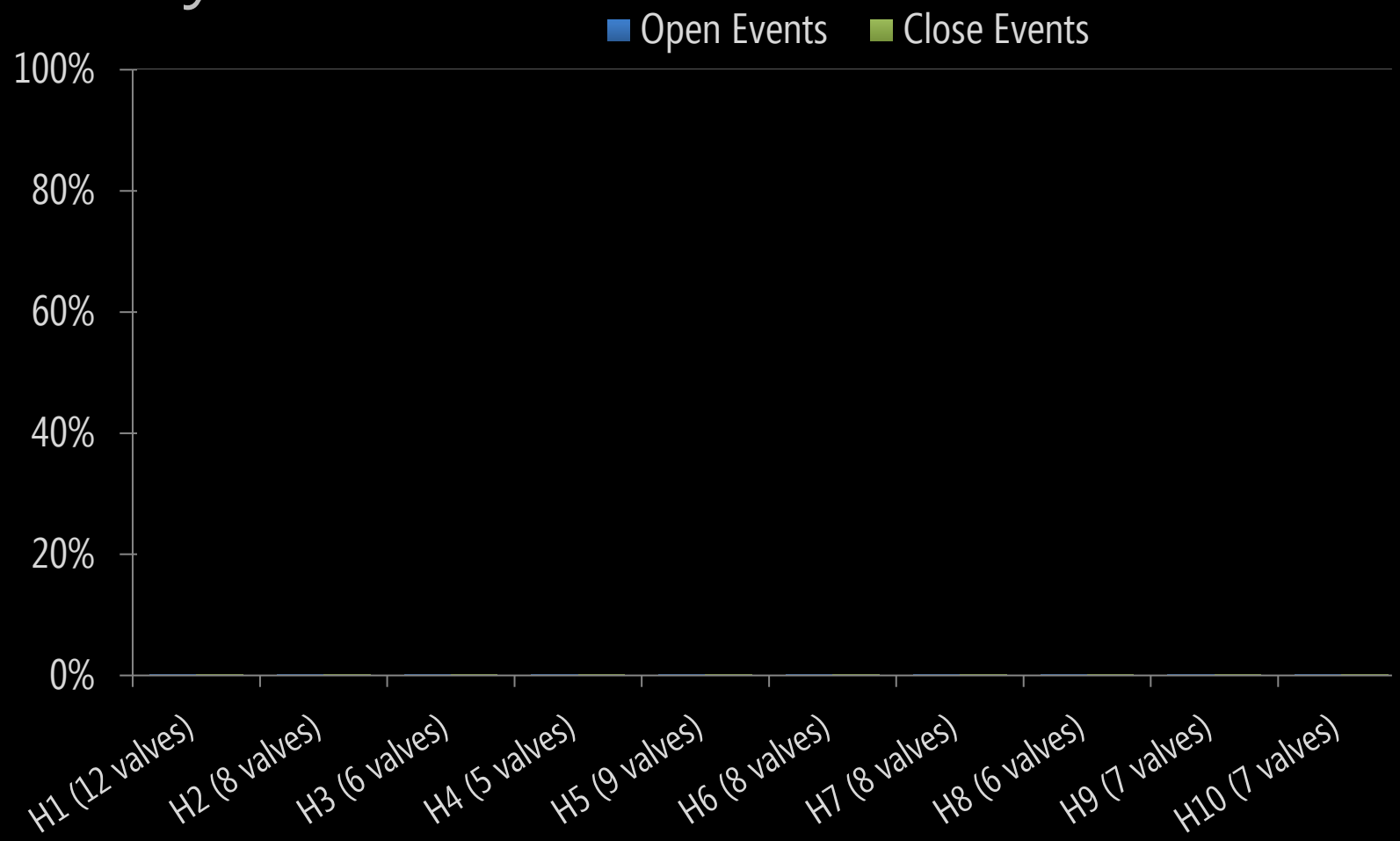
84 total fixtures

CLASSIFICATION EXPERIMENTS: 10-FOLD CROSS VALIDATION

1. Break data into 10 sets of size n/10
2. “Train” on 9 datasets and test on 1
3. Repeat for each combination of splits
4. Take mean accuracy

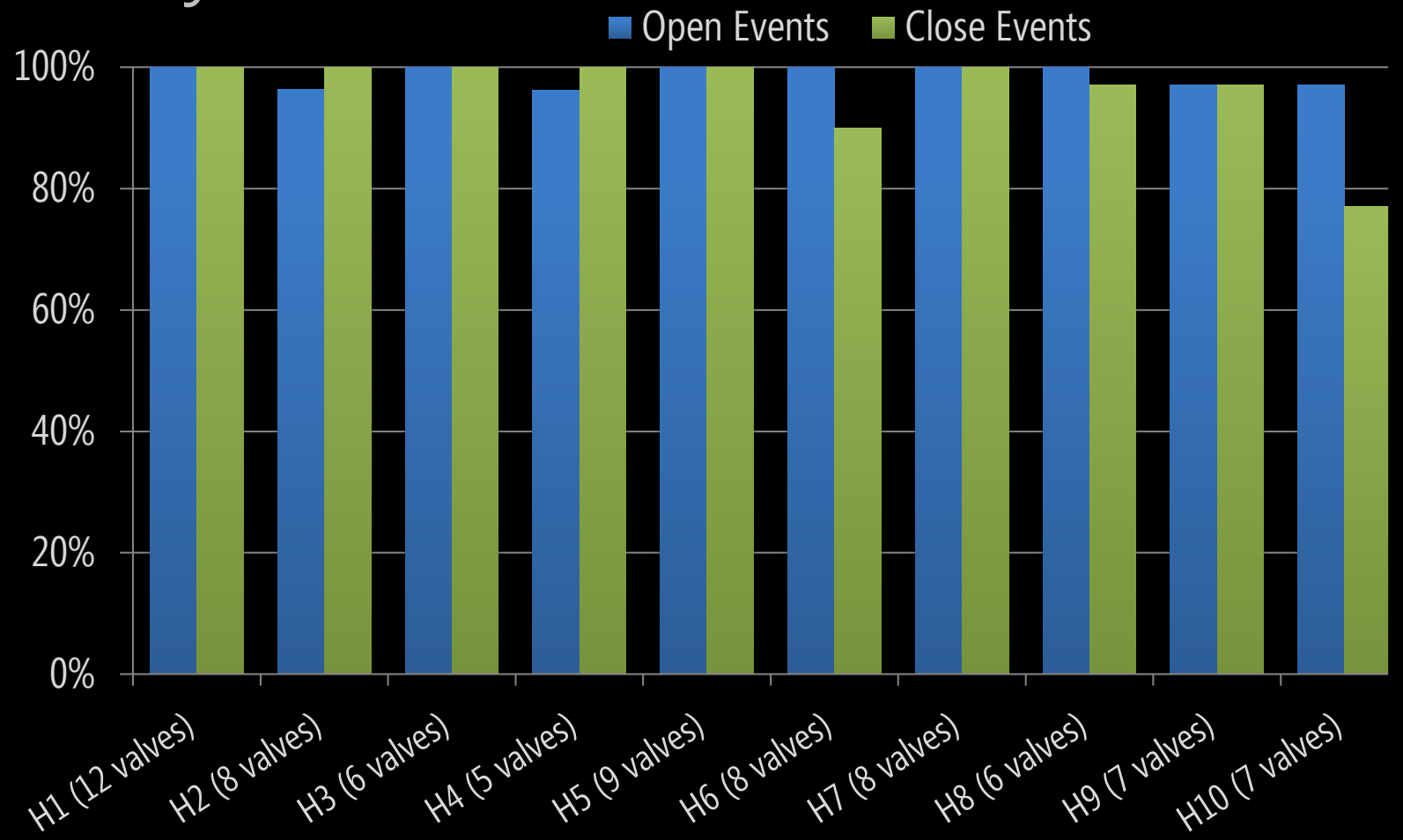
fixture classification results

by home



10-fold cross validation

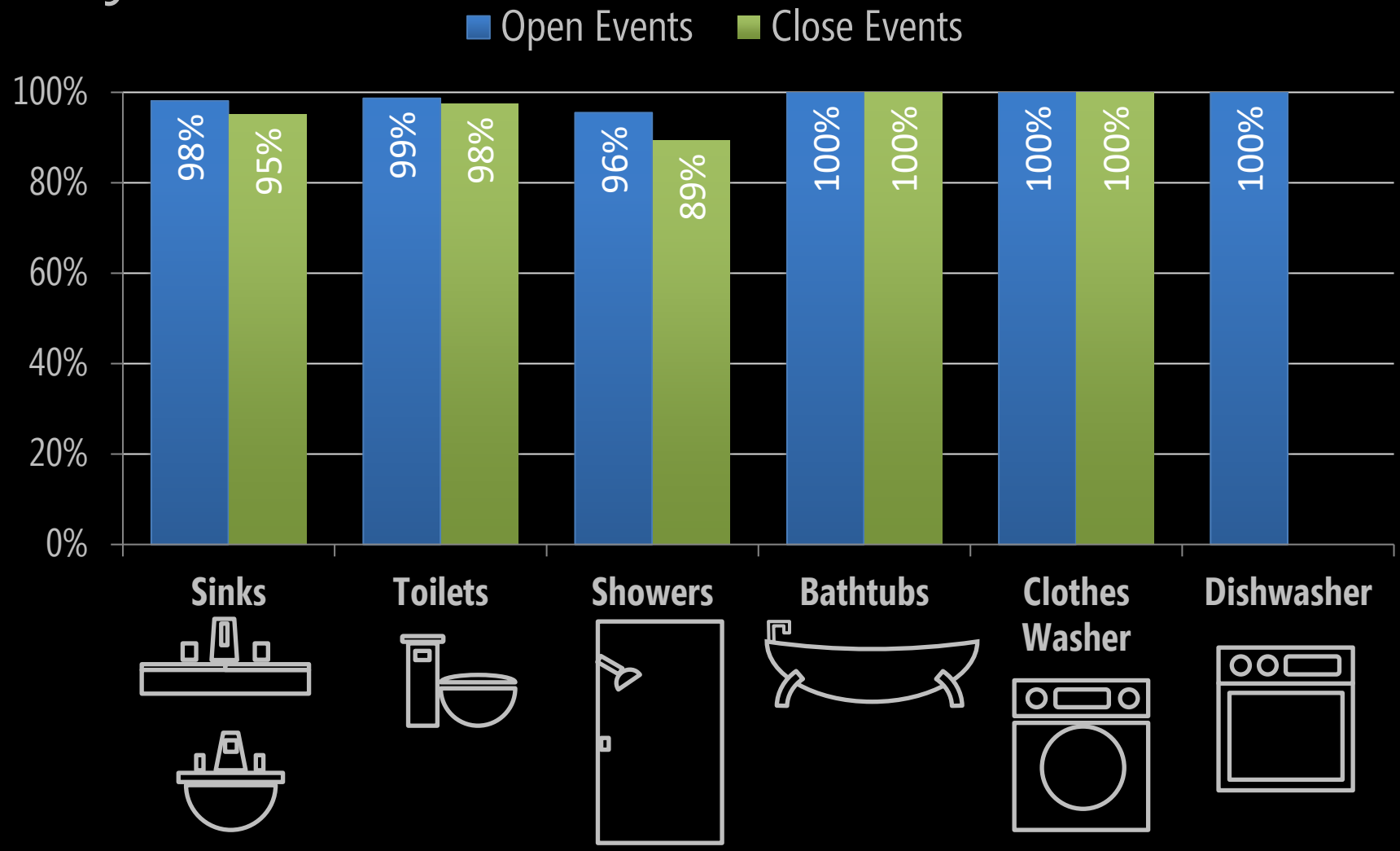
fixture classification results
by home



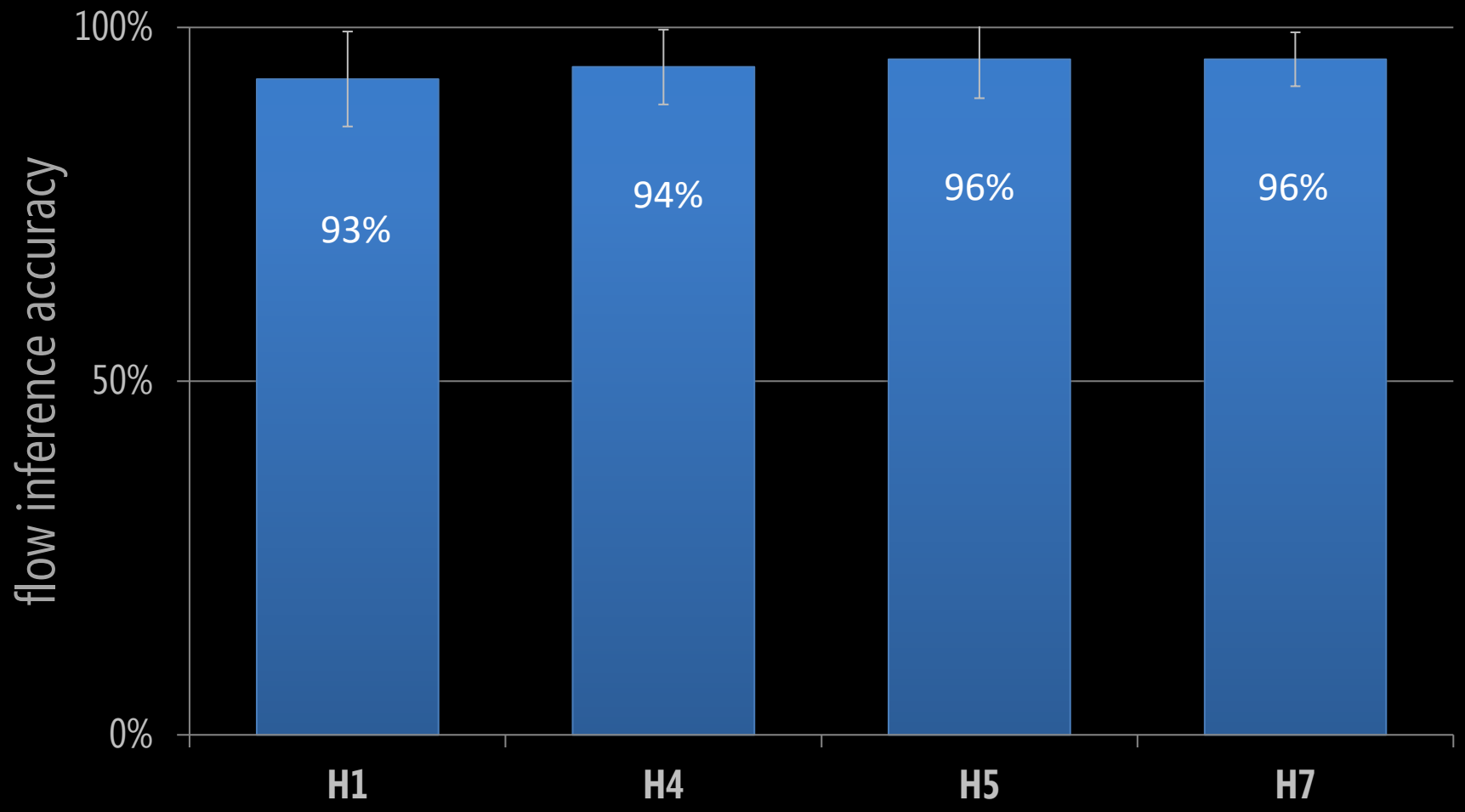
10-fold cross validation

fixture classification results

by fixture



flow inference results
by home



Within tolerances of domestic water meter accuracy; see [Arregui, 2003]

SP + ML IN HCI/UBICOMP

1. **Start by capturing signals under controlled conditions** with "ground truth" (*e.g.*, walk ten steps, wait, walk ten steps, wait). Collect easy, medium, and hard examples.

2. **Use tools to visualize, study, & process the signal** (both in time space and frequency space). Identify patterns. Brainstorm potential approaches for analysis. Calculate descriptive stats (*e.g.*, mean, median) both in time space & freq space.

3. **Search for previous solutions** to comparable signal processing/classification problems (*e.g.*, Google Scholar). What aspects can you re-appropriate for your work?

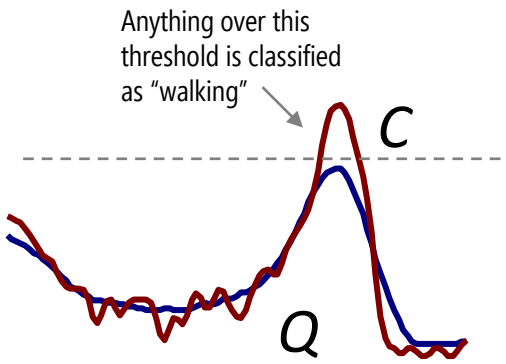
4. **Generate** and **test approaches offline** using test data. Iterate.

5. When satisfied with offline performance, **adapt approach to perform in real-time**. If real-time performance does not meet expectations, collect larger, more naturalistic test datasets and repeat process.

INTRO TO ML

3 PREVAILING APPROACHES FOR SIGNAL CLASSIFICATION

1. Rule-Based



2. Shape-Matching

3. Feature-Based

Raw Signal	1st Derivative	FFT
Mean	Mean	Top N freqs
Variance	Variance	Fundamental
Skewness	Skewness	
Kurtosis	Kurtosis	

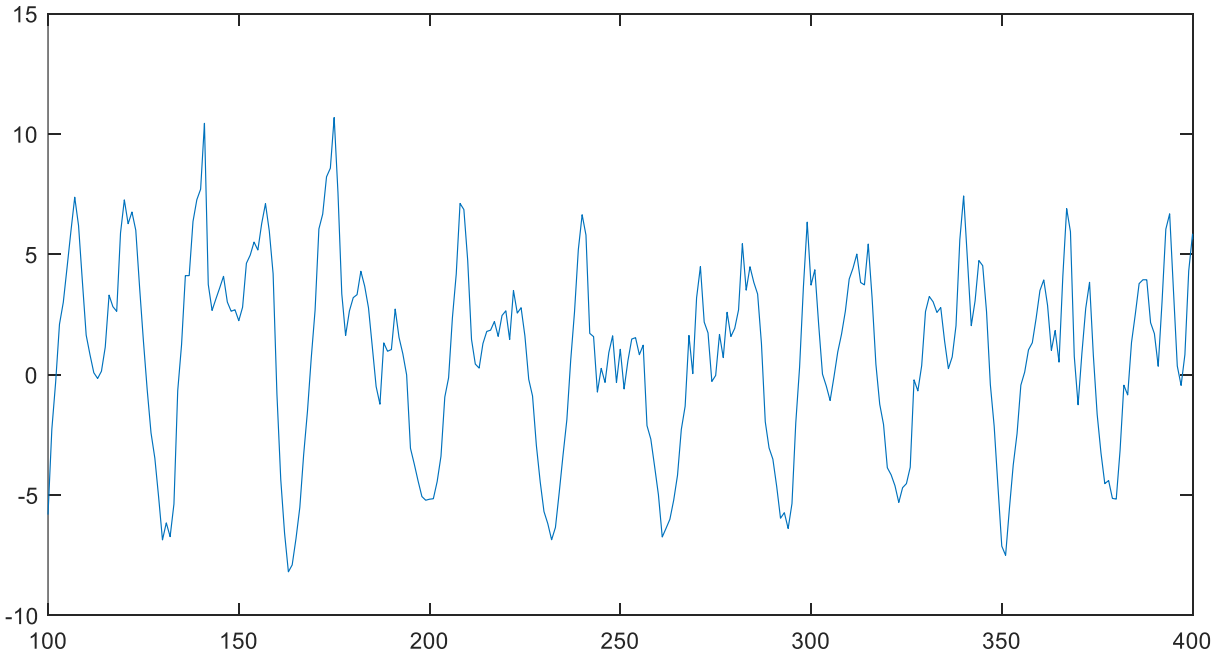
Raw Signal	1st Derivative	FFT
Mean	Mean	Top N freqs
Variance	Variance	Fundamental
Skewness	Skewness	
Kurtosis	Kurtosis	

Supervised Learning Model

Output

STEP TRACKER EXERCISE

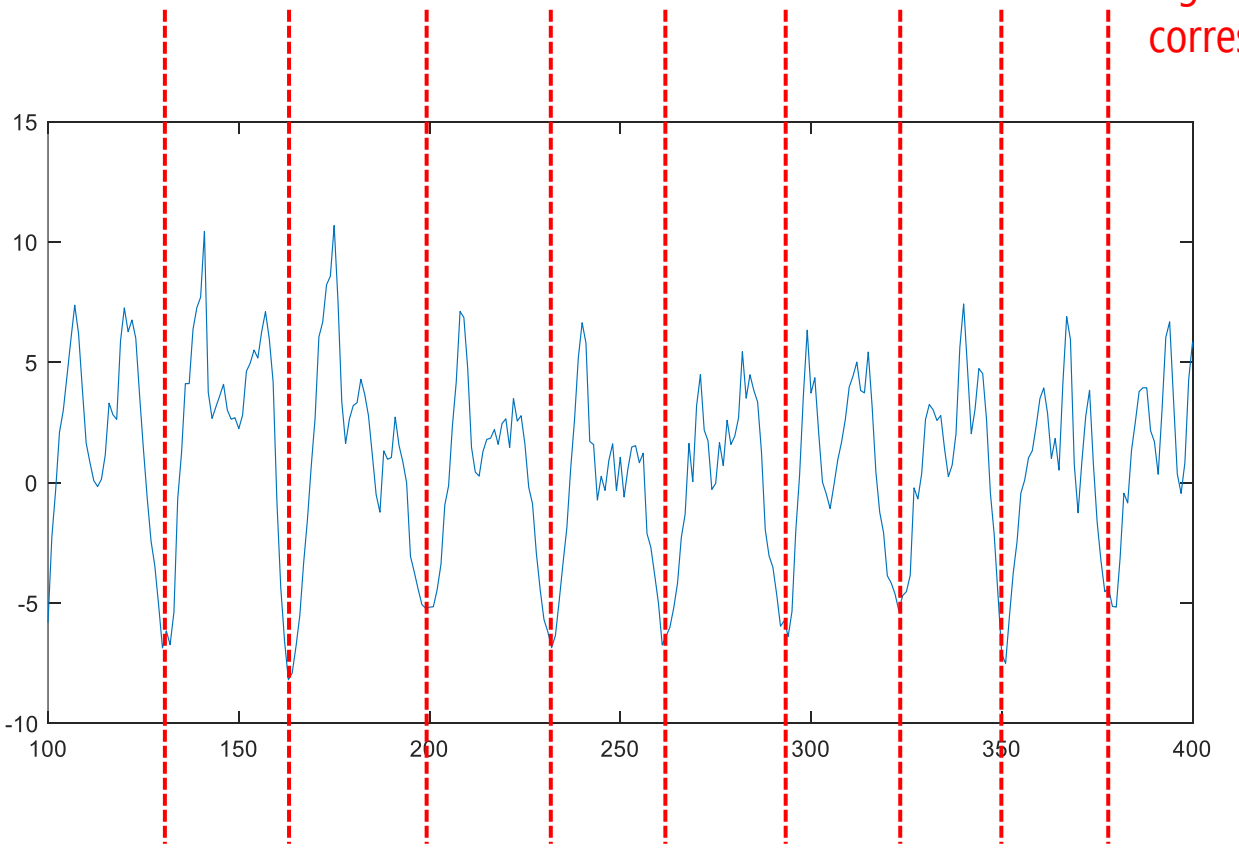
STEP TRACKER ACCELERATION SIGNAL



Think, Pair, Share

Take out a piece of paper and analyze this signal. Brainstorm and write down patterns that you observe. Then share them with a partner and discuss.

STEP TRACKER ACCELERATION SIGNAL



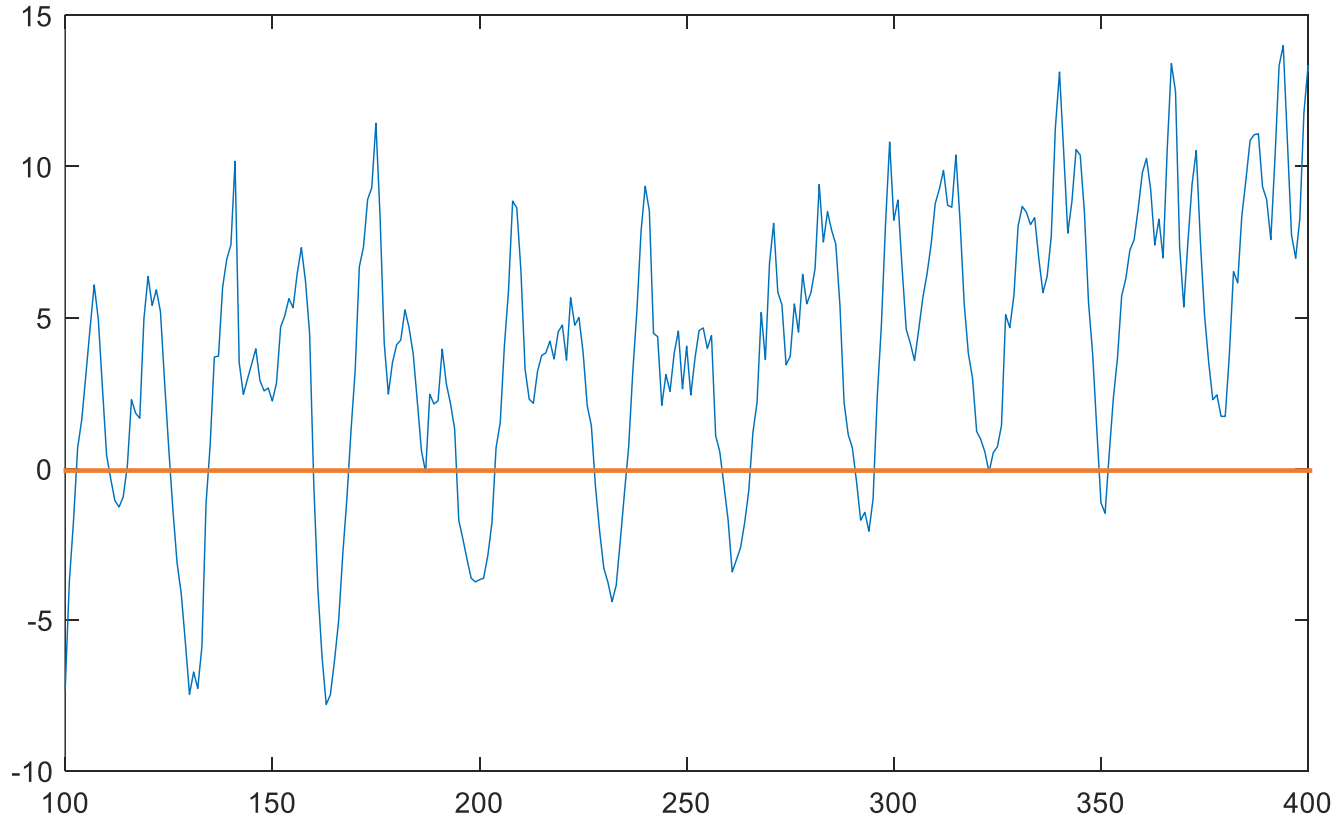
Signal repetition. Each cycle corresponds to a step

Think, Pair, Share

Take out a piece of paper and analyze this signal. Brainstorm and write down patterns that you observe. Then share them with a partner and discuss.

STEP TRACKER EXERCISE

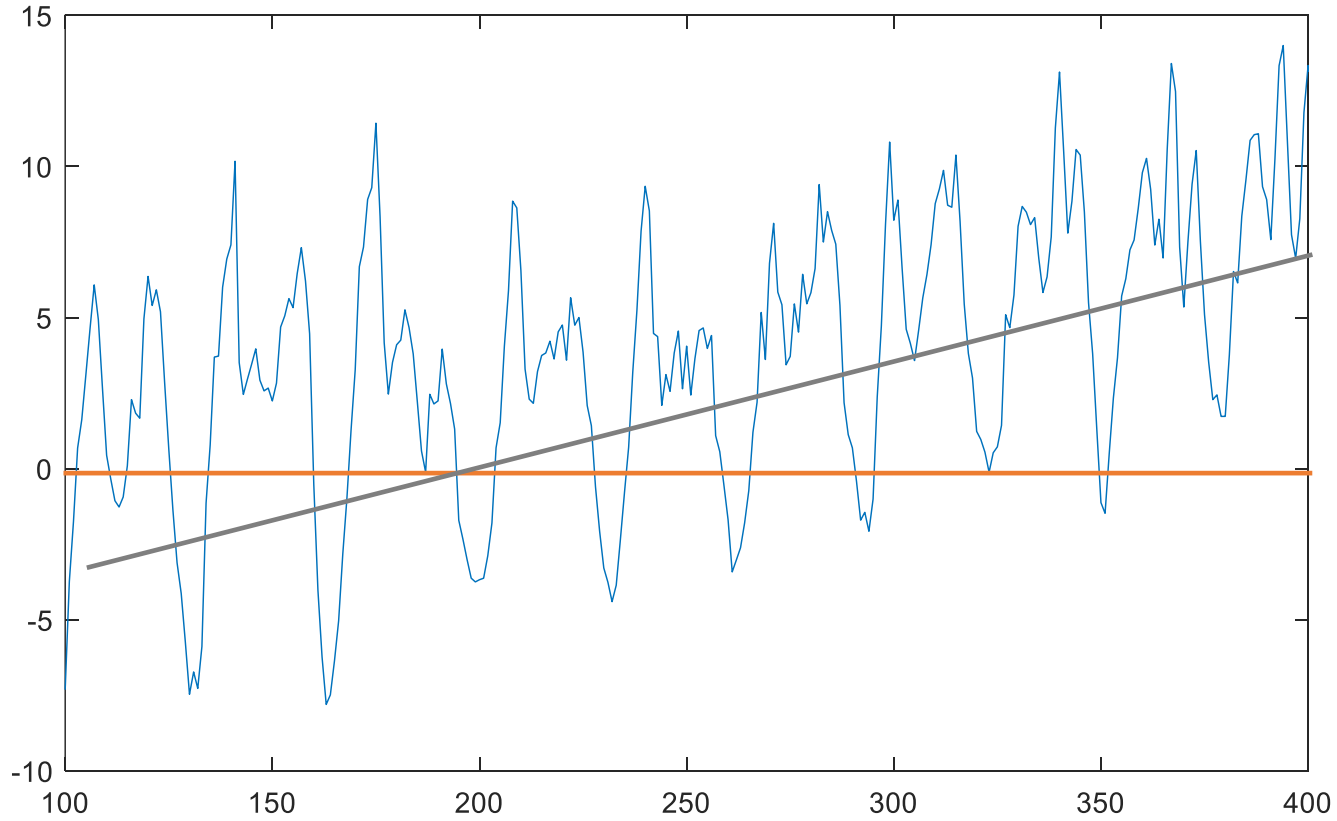
ONE WAY TO COUNT STEPS: DETECT ZERO CROSSINGS?



What's a potential problem with this?
Drift. The signal trends upwards.

STEP TRACKER EXERCISE

ONE WAY TO COUNT STEPS: DETECT ZERO CROSSINGS?

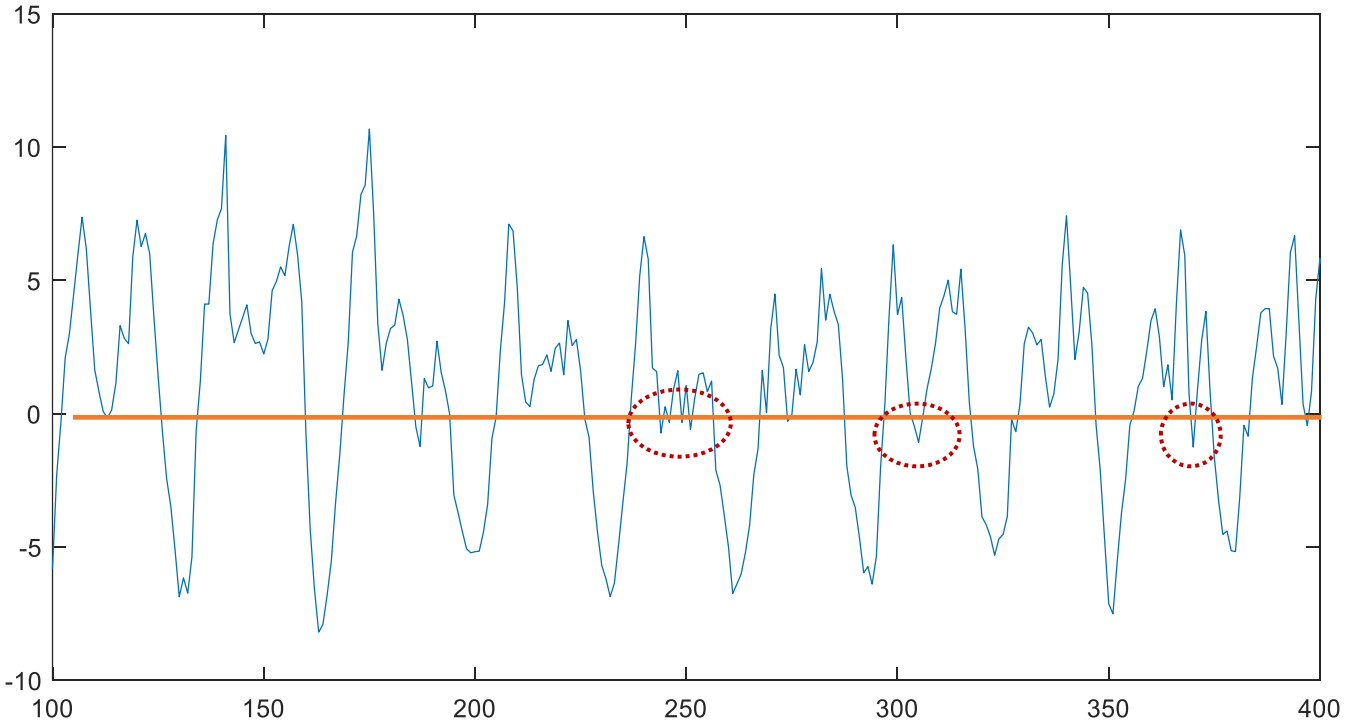


What's a potential problem with this?
Drift. The signal trends upwards.

Solution
Detrend the data. Subtract the mean or a best-fit line (like a regression line) from the data.

STEP TRACKER EXERCISE

ONE WAY TO COUNT STEPS: DETECT ZERO CROSSINGS?

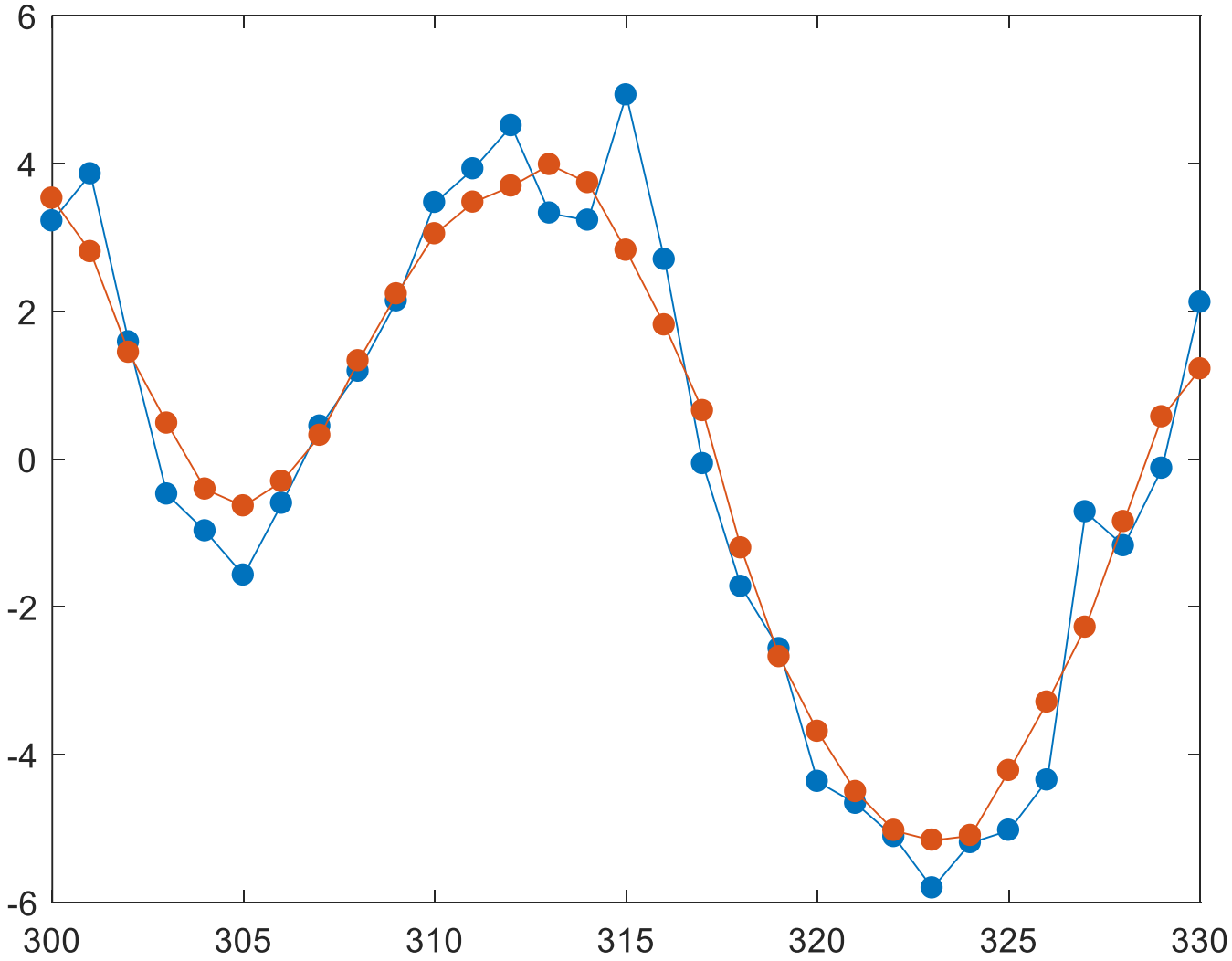


OK, this is better, but...
There are false positives.

Solution
Smooth the signal. The simplest is a moving average filter (which also serves as a cheap low-pass filter, that is it removes high frequency noise in your signal)

STEP TRACKER EXERCISE

MOVING AVERAGE FILTER

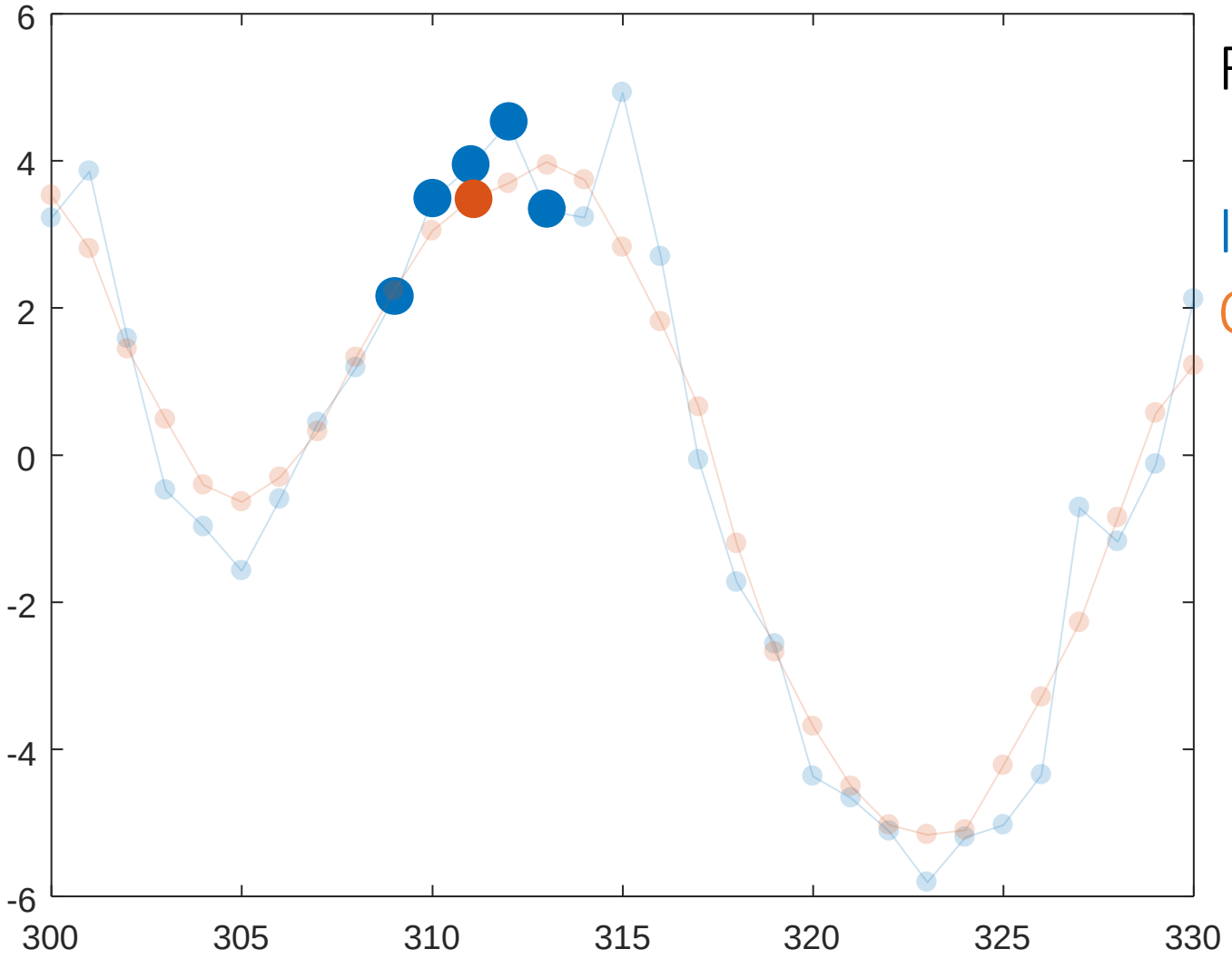


Noisy Signal

Filtered Signal

STEP TRACKER EXERCISE

MOVING AVERAGE FILTER



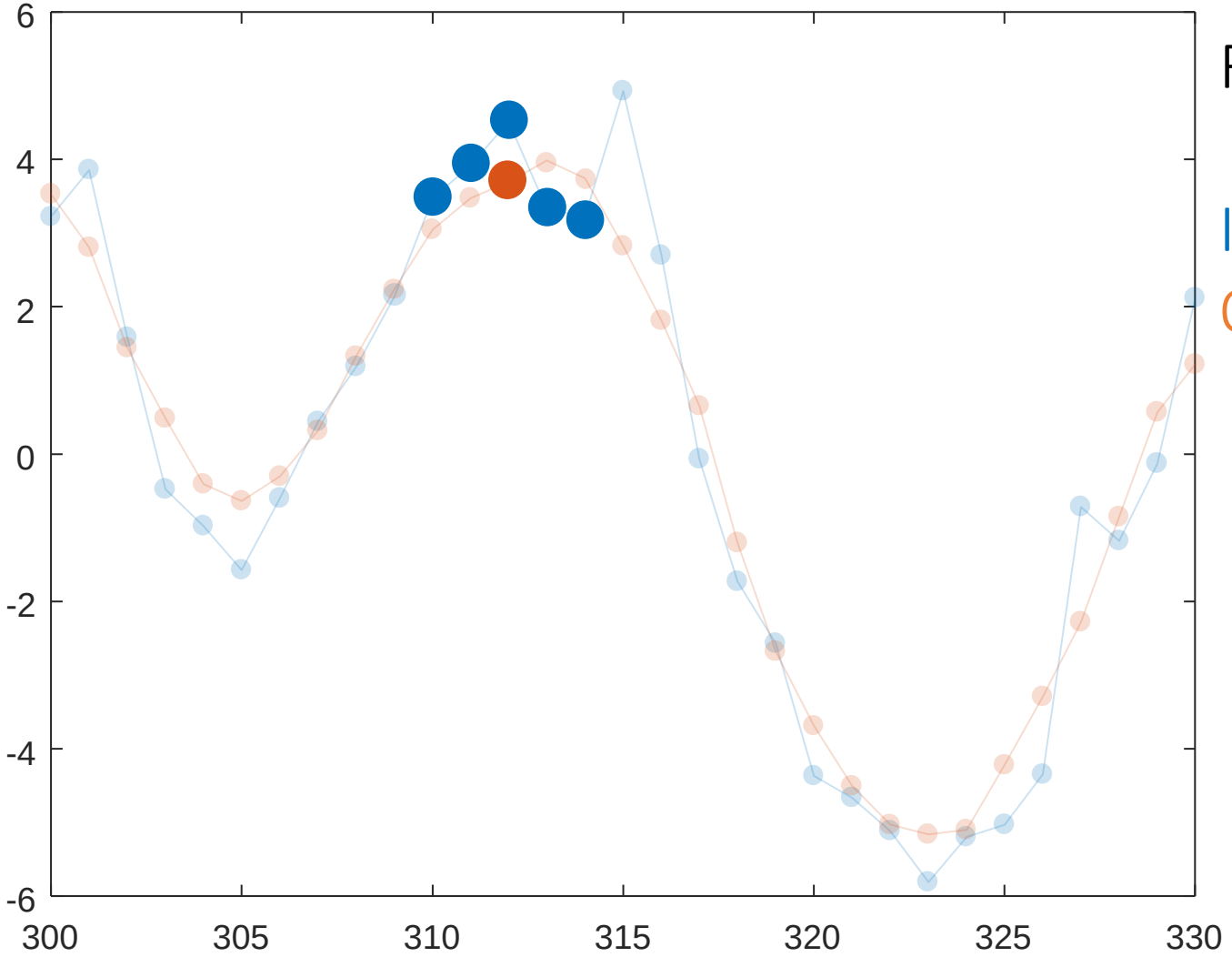
Filter Length [N] = 5

Input = [2.4,4,4.5,5,3.5]

Output = (2.4+4+4.5+5+3.5)/5 = 3.9

STEP TRACKER EXERCISE

MOVING AVERAGE FILTER



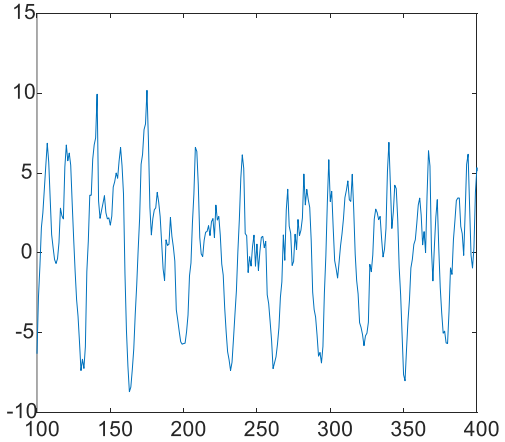
Filter Length [N] = 5

Input = [4,4.2,5,3.5,3.2]

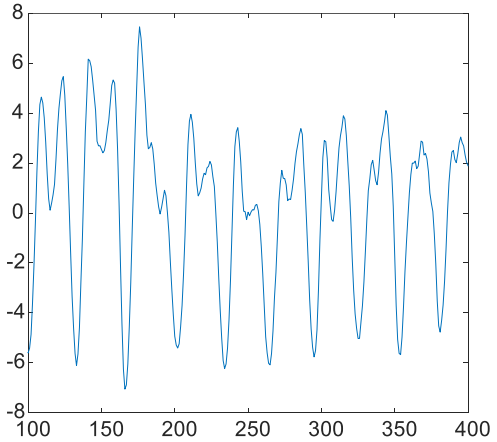
Output = (4+4.2+5+3.5+3.2)/5 = 4.0

STEP TRACKER EXERCISE

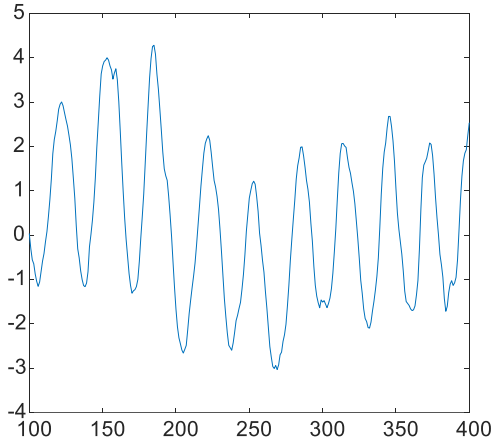
MOVING AVERAGE FILTER: DIFFERENT WINDOW SIZES



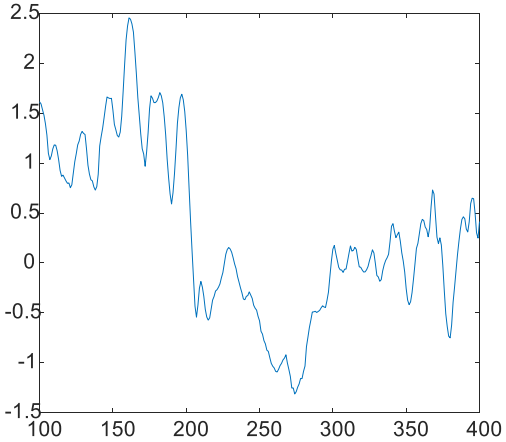
N = 1



N = 5

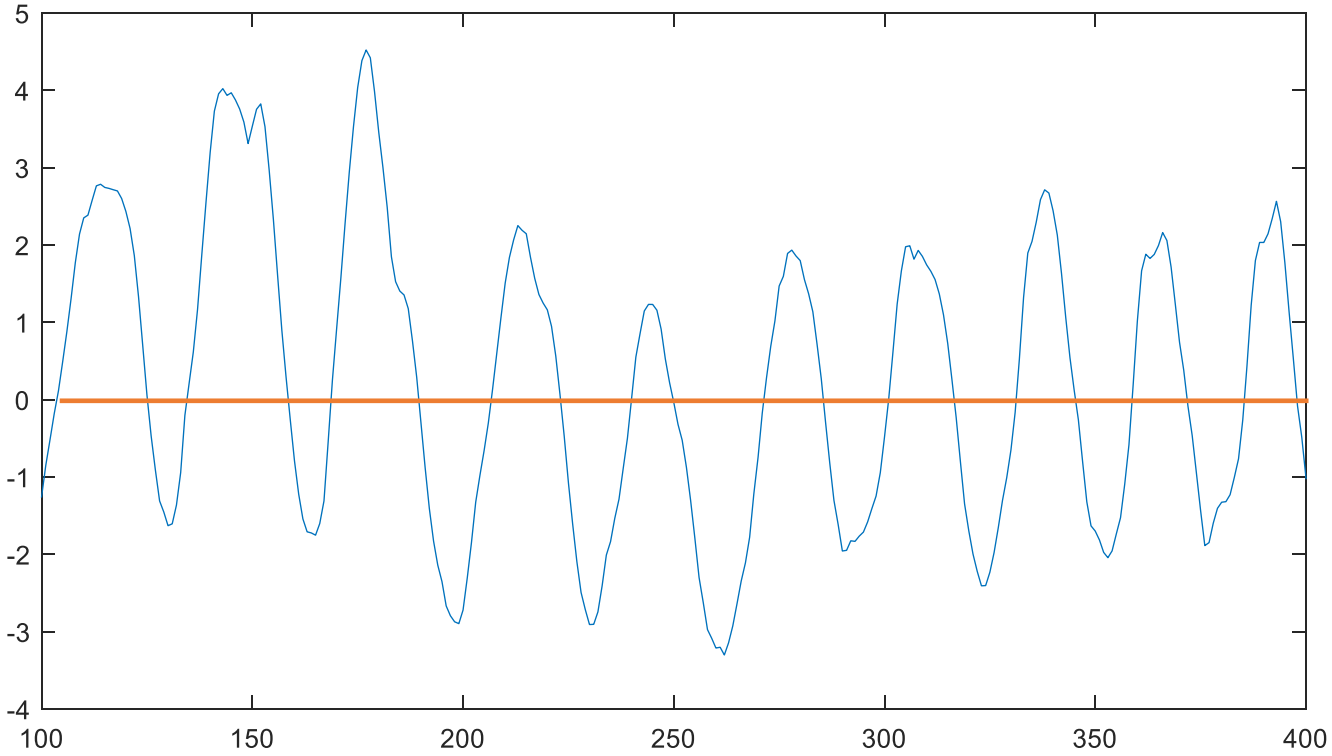


N = 15



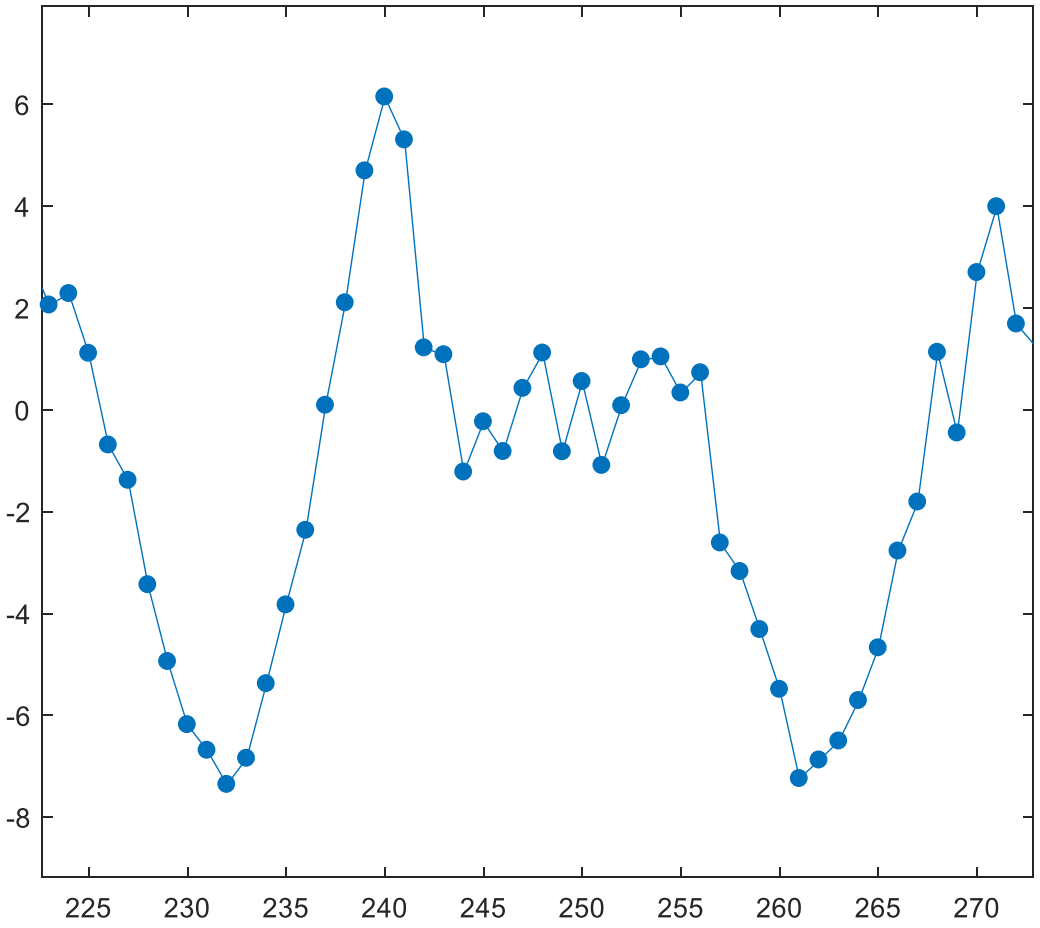
N = 30

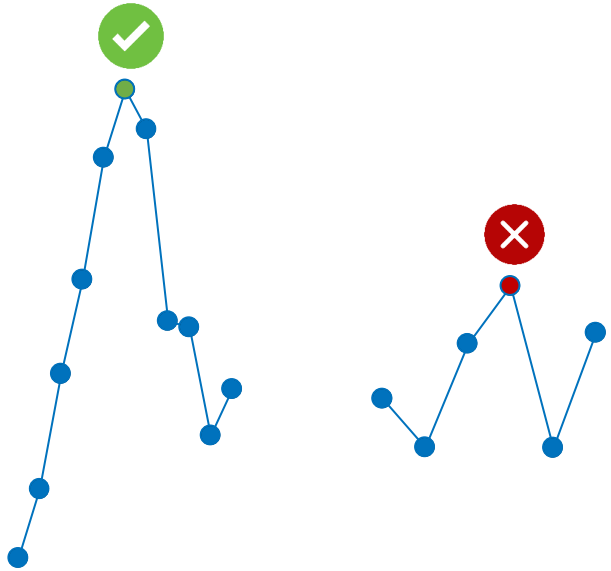
SMOOTHED SIGNAL: HOW TO DETECT STEPS?



Smoothed signal.

ONE WAY TO COUNT STEPS: DETECT PEAKS?



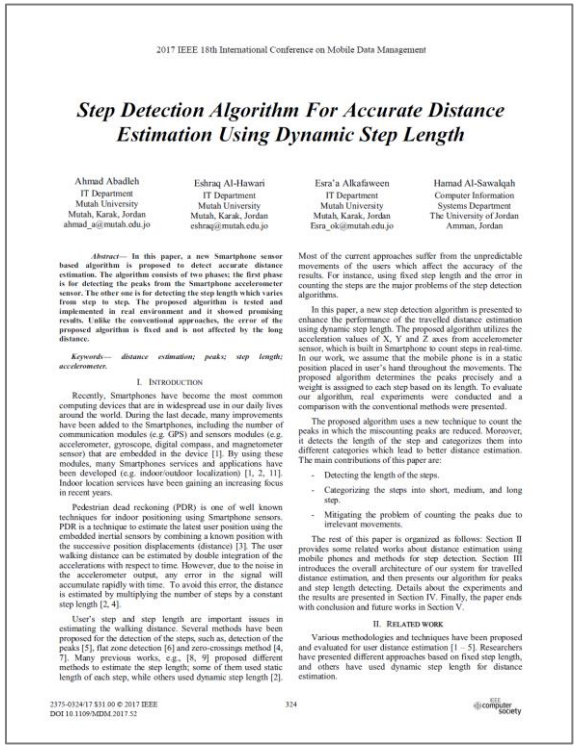


Final Think, Pair, Share

Take out a piece of paper and try to come up with a peak finding algorithm. I'll announce when it's time to share and discuss with a partner.

SMARTPHONE STEP TRACKING USING ACCELERATION

RELATED WORK: MANY USE A HEURISTIC APPROACH



2017 IEEE 18th International Conference on Mobile Data Management

Step Detection Algorithm For Accurate Distance Estimation Using Dynamic Step Length

Ahmad Abadleh, IT Department, Mutah University, Mutah, Karak, Jordan, ahmad_a@mutah.edu.jo
Eshraq Al-Hawari, IT Department, Mutah University, Mutah, Karak, Jordan, eshraq@mutah.edu.jo
Esra'a Alkafaween, IT Department, Mutah University, Mutah, Karak, Jordan, Esra_ok@mutah.edu.jo
Hamad Al-Sawalqah, Computer Information Systems Department, The University of Jordan, Amman, Jordan

Abstract— **In this paper, a new Smartphone sensor based algorithm is proposed to detect accurate distance estimation. The algorithm consists of two phases; the first phase is for detecting the peaks from the Smartphone accelerometer sensor. The other one is for detecting the step length which varies from step to step. The proposed algorithm is tested and implemented in real environment and it showed promising results. Unlike the conventional approaches, the error of the proposed algorithm is fixed and is not affected by the long distance.**

Keywords— distance estimation; peaks; step length; accelerometer.

I. INTRODUCTION

Recently, Smartphones have become the most common computing devices that are in widespread use in our daily lives around the world. During the last decade, many improvements have been added to the Smartphones, including the number of communication modules (e.g. GPS) and sensors modules (e.g. accelerometer, gyroscope, digital compass, and magnetometer sensor) that are embedded in the device [1]. By using these modules, many Smartphones services and applications have been developed (e.g. indoor/outdoor localization) [1, 2, 11]. Indoor location services have been gaining an increasing focus in recent years.

Pedestrian dead reckoning (PDR) is one of well known techniques for indoor positioning using Smartphone sensors. PDR is a technique to estimate the latest user position using the embedded inertial sensors by combining a known position with the successive position displacements (distance) [3]. The user walking distance can be estimated by double integration of the accelerations with respect to time. However, due to the noise in the accelerometer output, any error in the signal will accumulate rapidly with time. To avoid this error, the distance is estimated by multiplying the number of steps by a constant step length [2, 4].

User's step and step length are important issues in estimating the walking distance. Several methods have been proposed for the detection of the steps, such as, detection of the peaks [5], flat zone detection [6] and zero-crossings method [4, 7]. Many previous works, e.g., [8, 9] proposed different methods to estimate the step length; some of them used static length of each step, while others used dynamic step length [2]. Most of the current approaches suffer from the unpredictable movements of the users which affect the accuracy of the results. For instance, using fixed step length and the error in counting the steps are the major problems of the step detection algorithms.

In this paper, a new step detection algorithm is presented to enhance the performance of the travelled distance estimation using dynamic step length. The proposed algorithm utilizes the acceleration values of X, Y and Z axes from accelerometer sensor, which is built in Smartphone to count steps in real-time. In our work, we assume that the mobile phone is in a static position placed in user's hand throughout the movements. The proposed algorithm determines the peaks precisely and a weight is assigned to each step based on its length. To evaluate our algorithm, real experiments were conducted and a comparison with the conventional methods were presented.

The proposed algorithm uses a new technique to count the peaks in which the miscounting peaks are reduced. Moreover, it detects the length of the step and categorizes them into different categories which lead to better distance estimation. The main contributions of this paper are:

- Detecting the length of the steps.
- Categorizing the steps into short, medium, and long step.
- Mitigating the problem of counting the peaks due to irrelevant movements.

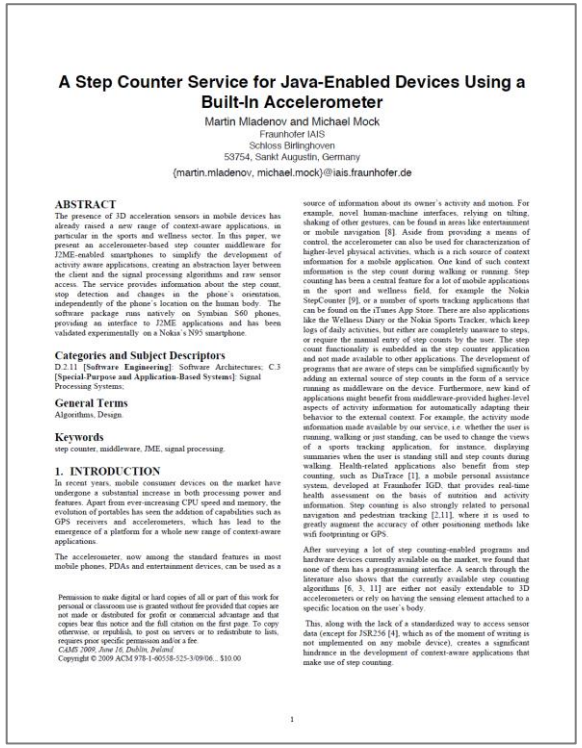
The rest of this paper is organized as follows: Section II provides some related works about distance estimation using mobile phones and methods for step detection. Section III introduces the overall architecture of our system for travelled distance estimation, and then presents our algorithm for peaks and step length detecting. Details about the experiments and the results are presented in Section IV. Finally, the paper ends with conclusion and future works in Section V.

II. RELATED WORK

Various methodologies and techniques have been proposed and evaluated for user distance estimation [1 – 5]. Researchers have presented different approaches based on fixed step length, and others have used dynamic step length for distance estimation.

2375-0324/17 $31.00 © 2017 IEEE
DOI 10.1109/MDM.2017.52

Step Detection Algorithm for Accurate Distance Estimation Using Dynamic Step Length
Abadleh *et al.,* IEEE MDM'17



A Step Counter Service for Java-Enabled Devices Using a Built-In Accelerometer

Martin Mladenov and Michael Mock
Fraunhofer IAIS
Schloss Birlinghoven
53754, Sankt Augustin, Germany
{martin.mladenov, michael.mock}@iais.fraunhofer.de

ABSTRACT

The presence of 3D acceleration sensors in mobile devices has already raised a new range of context-aware applications, in particular in the sports and wellness sector. In this paper, we present an accelerometer-based step counter middleware for J2ME-enabled smartphones to simplify the development of activity aware applications, creating an abstraction layer between the client and the signal processing algorithms and raw sensor access. The service provides information about the step count, stop detection and changes in the phone's orientation, independently of the phone's location on the human body. The software package runs natively on Symbian S60 phones, providing an interface to J2ME applications and has been validated experimentally on a Nokia's N95 smartphone.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures; C.3 [**Special-Purpose and Application-Based Systems**]: Signal Processing Systems;

General Terms

Algorithms, Design.

Keywords

step counter, middleware, JME, signal processing.

1. INTRODUCTION

In recent years, mobile consumer devices on the market have undergone a substantial increase in both processing power and features. Apart from ever-increasing CPU speed and memory, the evolution of portables has seen the addition of capabilities such as GPS receivers and accelerometers, which has lead to the emergence of a platform for a whole new range of context-aware applications.

The accelerometer, now among the standard features in most mobile phones, PDAs and entertainment devices, can be used as a source of information about its owner's activity and motion. For example, novel human-machine interfaces, relying on tilting, shaking of other gestures, can be found in areas like entertainment or mobile navigation [8]. Aside from providing a means of control, the accelerometer can also be used for characterization of higher-level physical activities, which is a rich source of context information for a mobile application. One kind of such context information is the step count during walking or running. Step counting has been a central feature for a lot of mobile applications in the sport and wellness field, for example the Nokia StepCounter [9], or a number of sports tracking applications that can be found on the iTunes App Store. There are also applications like the Wellness Diary or the Nokia Sports Tracker, which keep logs of daily activities, but either are completely unaware to steps, or require the manual entry of step counts by the user. The step count functionality is embedded in the step counter application and not made available to other applications. The development of programs that are aware of steps can be simplified significantly by adding an external source of step counts in the form of a service running as middleware on the device. Furthermore, new kind of applications might benefit from middleware-provided higher-level aspects of activity information for automatically adapting their behavior to the external context. For example, the activity mode information made available by our service, i.e. whether the user is running, walking or just standing, can be used to change the views of a sports tracking application, for instance, displaying summaries when the user is standing still and step counts during walking. Health-related applications also benefit from step counting, such as DiaTrace [1], a mobile personal assistance system, developed at Fraunhofer IGD, that provides real-time health assessment on the basis of nutrition and activity information. Step counting is also strongly related to personal navigation and pedestrian tracking [2,11], where it is used to greatly augment the accuracy of other positioning methods like wifi fingerprinting or GPS.

After surveying a lot of step counting-enabled programs and hardware devices currently available on the market, we found that none of them has a programming interface. A search through the literature also shows that the currently available step counting algorithms [6, 3, 11] are either not easily extendable to 3D accelerometers or rely on having the sensing element attached to a specific location on the user's body.

This, along with the lack of a standardized way to access sensor data (except for JSR256 [4], which as of the moment of writing is not implemented on any mobile device), creates a significant hindrance in the development of context-aware applications that make use of step counting.

CAMS'2009, June 16, Dublin, Ireland

A Step Counter Service for Java-Enabled Devices Using a Built-in Accelerometer
Mladenov and Mock, CAMS'09

A high-accuracy step counting algorithm for iPhones using Accelerometer

Kinh Tran, Tu Le and Tien Dinh
Faculty of Information Technology, University of Sciences, VNU-HCMC
227 Nguyen Van Cu, Ho Chi Minh City, Vietnam
tranvinhkinh1@yahoo.com
anhtu.khtn@gmail.com
dbtien@fit.hcmus.edu.vn

Abstract—Nowadays, smartphones become very popular in our daily activities. Current phone models have been integrated with high-accuracy built-in sensors. Many developers have been taking advantage of those sensors to build useful applications. Tracking user's activities via smartphones is one of the hot topics in research community. Step detection or step counting is one of the key problems in the field of understanding user's activities. There are a few research work and many commercial smartphone applications to detect or count the user's steps. In other work, researchers have tried to detect or count the steps by attaching the phone into a specific position. In this paper, we propose a new method based on Kalman Filter to detect and count the user's step with higher accuracy using built-in accelerometer in iOS. In our experiments, we allow users to hold the phone in hand or keep it in the pocket. The experimental results show that our approach gives better results than other commercial applications in the market.

Keywords—step detection,step counting, phone's sensor, user activity, accelerometer.

Fig. 1. Raw signals from accelerometer.

I. INTRODUCTION

Human activities such as walking, jumping, typing, and talking can be detected by one or a set of accelerometers attached into the body [1]. Recently, new smartphone models integrated with different kinds of sensors become popular in real life. In the last couple of years, with the introduction of iPhones into the market, stepping is the most common activity that we can detect with acceptable accuracy.

Our step detection method based on analysis of humans acceleration signal. Those signals are generated by changes on smartphone velocity and position which caused by humans acceleration. Therefore, humans step detection relies on the relative position between smartphone and human. In our work, experimenters are required holding the smartphone on hand with its display screen point upward (On-hand mode) or putting the smartphone in their pocket or pants pocket with the smartphone's top point upright or downright (Pocket mode) (see Experiment section for illustration).

The raw signal from accelerometer is a combination of gravity, humans acceleration, and a lot of noises. So, it is inaccuracy to detect step directly from this raw signal. Figure 1 show 3 acceleration graphs correspond to 3 axes in On-hand mode. From this figure, it is very difficult to count that there are 10 steps. For this reason, we use an existing filter and propose a method for step detecting. Our method includes two filters: Discrete Kalman filter [8] and our filter. In the experiment, we develop an application to count human step (pedometer) and we compare it with other pedometer applications from AppStore. We choose applications with high ratings on AppStore. As the result, our pedometer has average accuracy higher than other pedometers. And the average difference in number of steps is around 3 to 4 steps of each 100 steps.

The Method section describes in detail how we analyse signal from accelerometer, how we use Discrete Kalman filter and how our filter work. The Experiment section describes steps to do experiment and shows some results from it.

II. RELATED WORK

Step detection is not a new topic in research community, there are many related work on step detection, step counting or even user activity recognition. In 2004, there is a related work on user activity recognition: "Activity recognition from User-Annotated Acceleration Data" by Ling Bao and Stephen S.Intille [1]. This work showed an interesting result that their algorithm can recognize 20 activities using acceleration data annotated provided by 5 attached accelerometers on experimenters.

In 2008, Ryan Libby proposed a method for real-time step detection "A simple method for reliable footstep detection on embedded sensor platforms" [3]. He presents a method for reliable step detection by using 3-dimensional accelerometer. In his work, the accelerometer was worn clipped to clothing at the hip. His method works on step detection of bicycling, running or walking. However, his method is not accuracy when step periods fall out the middle range.

978-1-4673-5604-6/12/$31.00 ©2012 IEEE

A High-Accuracy Step Counting Algorithm for iPhones using Accelerometer
Tran, Le, & Dinh, IEEE ISSPIT'12

• • •

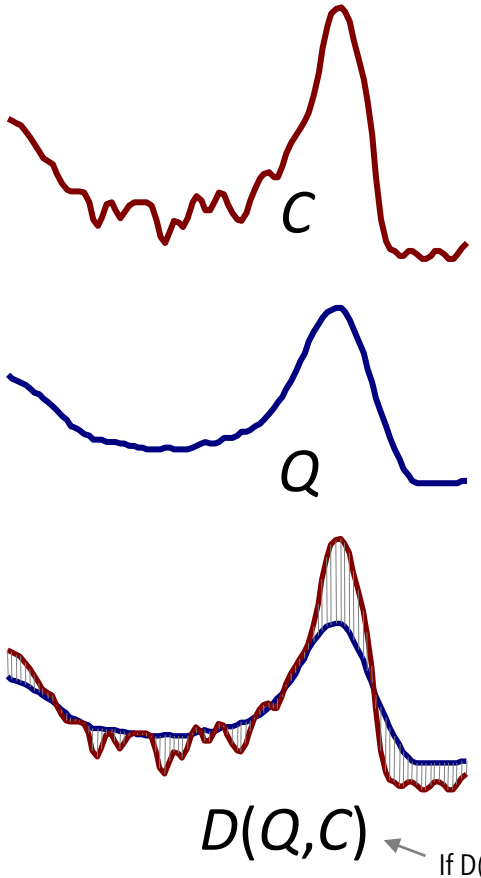
And many others…
Search for some one your own.
What are some commonalities?
What could you integrate into your solution?

INTRO TO ML

3 PREVAILING APPROACHES FOR SIGNAL CLASSIFICATION

1. Rule-Based

2. Shape-Matching



If D(Q,C) < threshold, we consider them the same signal

3. Feature-Based

Raw Signal	1st Derivative	FFT
Mean	Mean	Top N freqs
Variance	Variance	Fundamental
Skewness	Skewness	
Kurtosis	Kurtosis	

Raw Signal	1st Derivative	FFT
Mean	Mean	Top N freqs
Variance	Variance	Fundamental
Skewness	Skewness	
Kurtosis	Kurtosis	

Supervised Learning Model

Output

A3: Signal Processing + Machine Learning 1: Shape-Matching Gesture Recognizer

Published | Edit

Overview

Imagine working for a new hardware startup designing new input controllers. You've been asked to prototype a new, custom input device with gesture recognition (using an accelerometer)--for example, to use the input controller as a paddle in tennis, as a "ball" in bowling, or to recognize the "overhand throwing motion" in baseball. In this assignment, you will build your own 3D gesture recognizer to automatically recognize these gestures.

While in the "real world" you would ultimately need to create a real-time gesture recognizer, for this assignment you will make an *offline* version in Jupyter Notebook (we strongly recommend the Anaconda Distribution). Specifically, in this assignment you will build a *shape-matching* (or *template-matching*) recognizer such as via a Euclidean distance metric or Dynamic Time Warping. In A4, you will build a *feature-based* (or *model-based*) recognizer using a support-vector machine (SVM) (recommended) or an alternative supervised learning approach of your choosing (*e.g.*, an HMM).

Within Jupyter Notebook, we will use Python 3 and these amazing libraries numpy, scipy, matplotlib, and scikit-learn. Numpy and scipy provide numeric array handling and signal processing, matplotlib provides visualization, and scikit-learn is the de facto machine learning library in Python. You are welcome to use other libraries as well (*e.g.*, this DTW library).

For your deliverables, you will turn in your Jupyter Notebook, your recorded gestures, and a slide deck report on your algorithmic approaches and performance results.

Things You Need

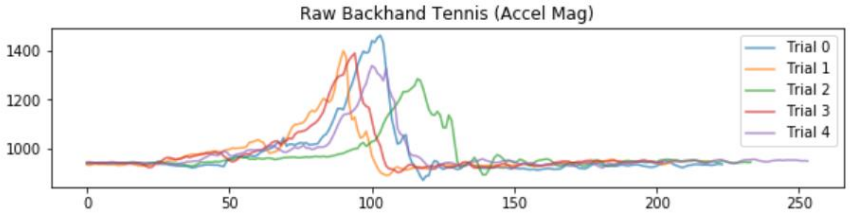
- Download and install Jupyter Notebook for offline data analysis, signal processing, and machine learning. We've created an initial skeleton to parse the data, which is available here.

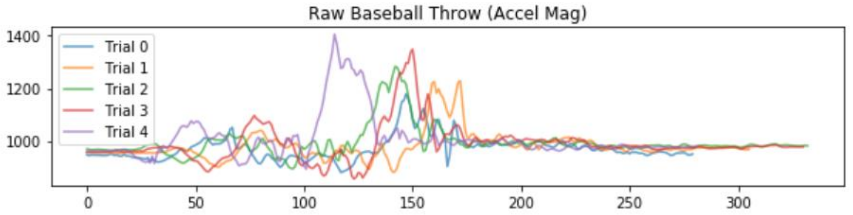
Related Items
SpeedGrader™

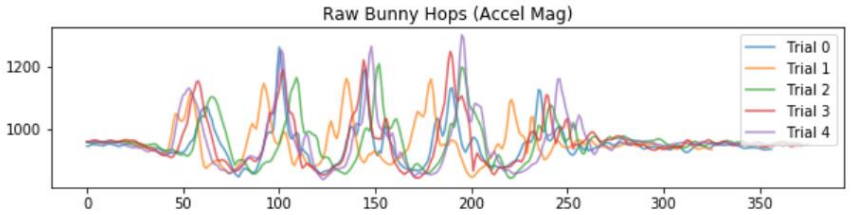
A3: SHAPE-BASED GESTURE RECOGNITION

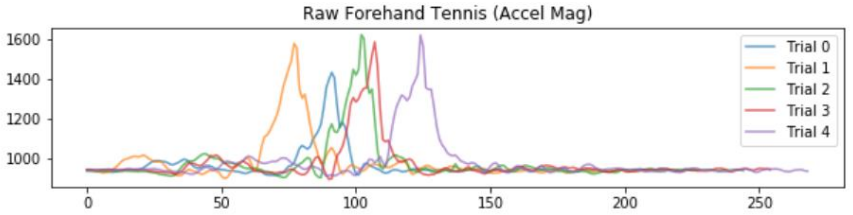
GENERAL APPROACH

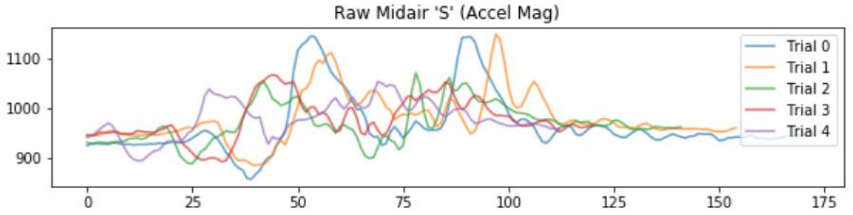
1. Visualize the data. Look for patterns. Brainstorm how to leverage those patterns in your algorithms

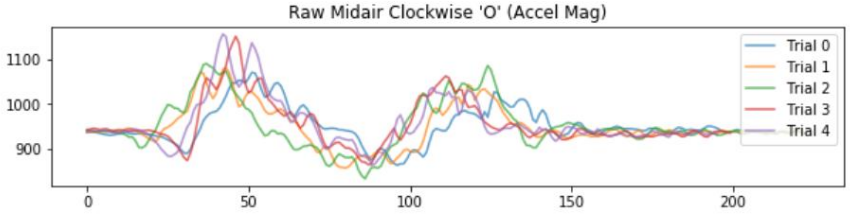


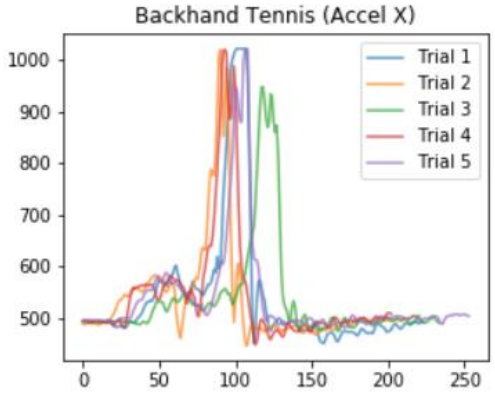


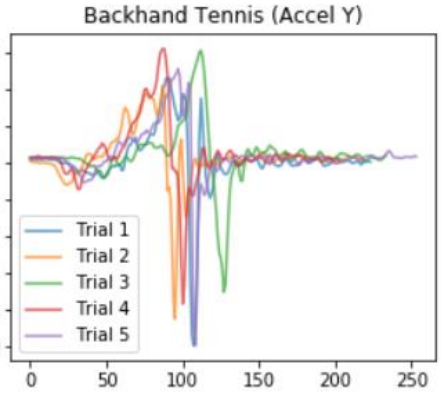


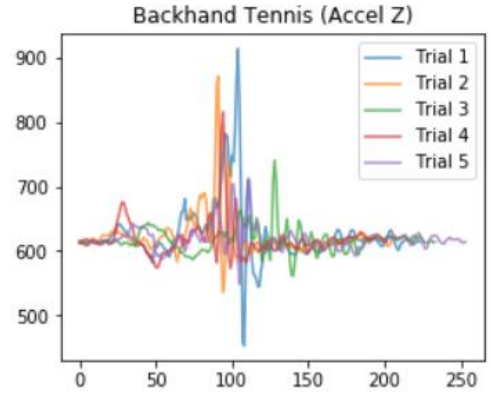


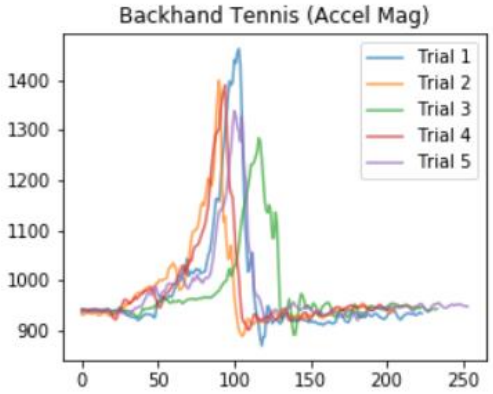


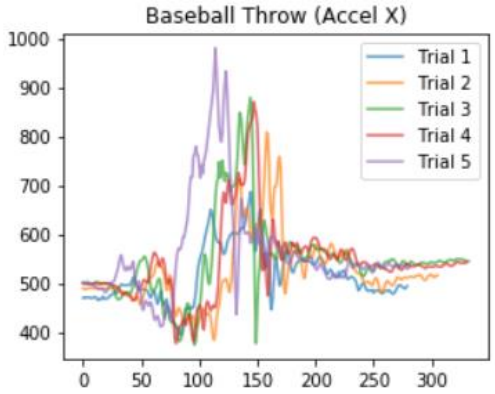


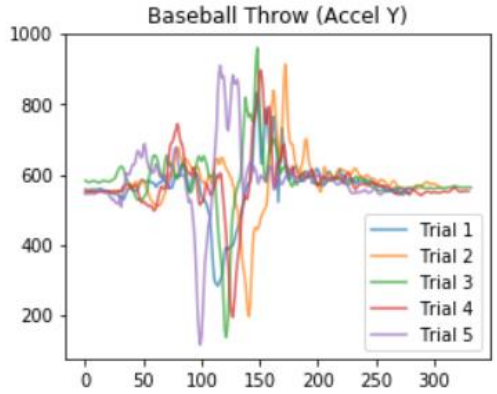

Backhand Tennis (Accel X)
Trial 1
Trial 2
Trial 3
Trial 4
Trial 5

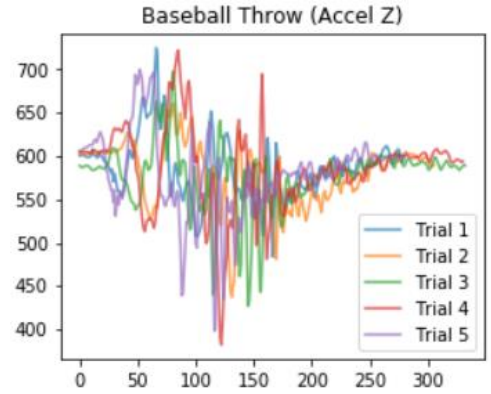

Backhand Tennis (Accel Y)
Trial 1
Trial 2
Trial 3
Trial 4
Trial 5

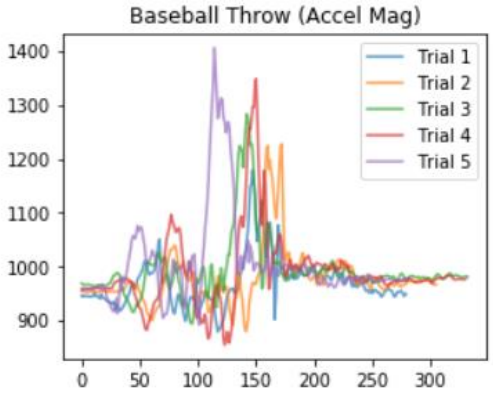

Backhand Tennis (Accel Z)
Trial 1
Trial 2
Trial 3
Trial 4
Trial 5

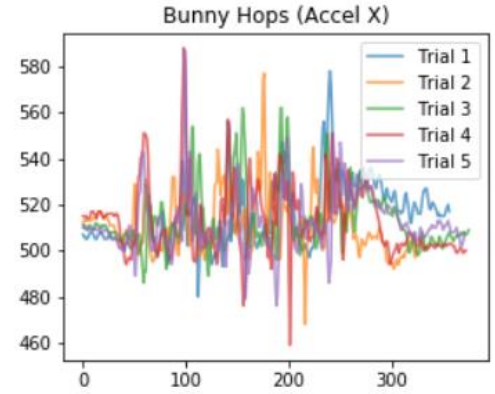

Backhand Tennis (Accel Mag)
Trial 1
Trial 2
Trial 3
Trial 4
Trial 5

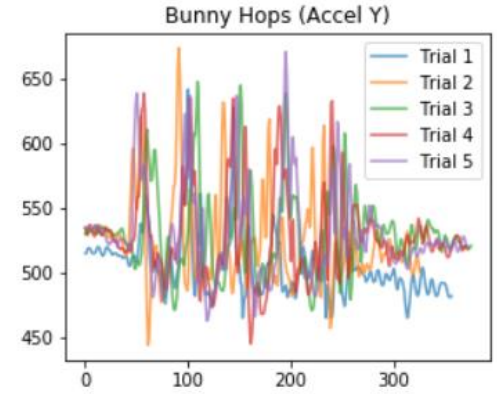

Baseball Throw (Accel X)
Trial 1
Trial 2
Trial 3
Trial 4
Trial 5

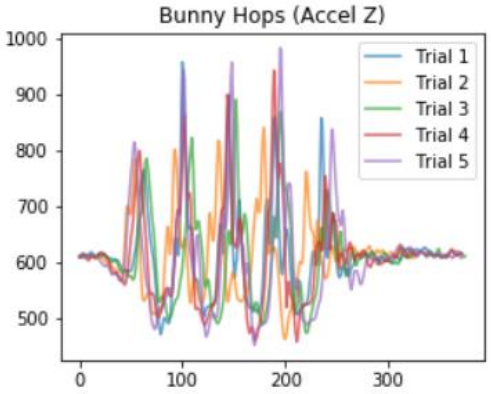

Baseball Throw (Accel Y)
Trial 1
Trial 2
Trial 3
Trial 4
Trial 5

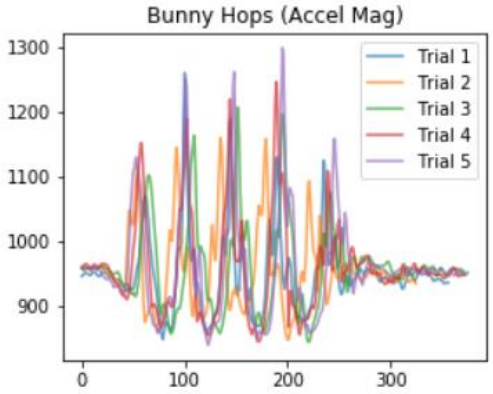

Baseball Throw (Accel Z)
Trial 1
Trial 2
Trial 3
Trial 4
Trial 5


Baseball Throw (Accel Mag)
Trial 1
Trial 2
Trial 3
Trial 4
Trial 5


Bunny Hops (Accel X)
Trial 1
Trial 2
Trial 3
Trial 4
Trial 5


Bunny Hops (Accel Y)
Trial 1
Trial 2
Trial 3
Trial 4
Trial 5


Bunny Hops (Accel Z)
Trial 1
Trial 2
Trial 3
Trial 4
Trial 5

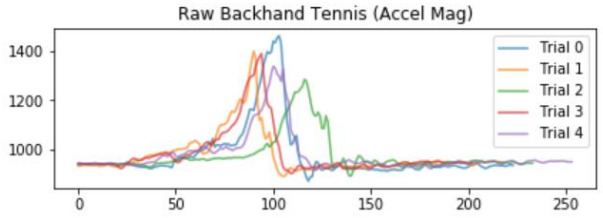

Bunny Hops (Accel Mag)
Trial 1
Trial 2
Trial 3
Trial 4
Trial 5

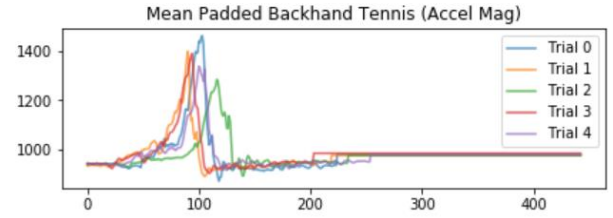
A3: SHAPE-BASED GESTURE RECOGNITION

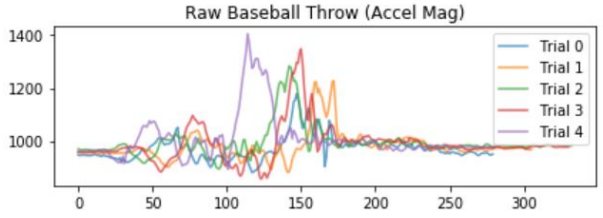
GENERAL APPROACH

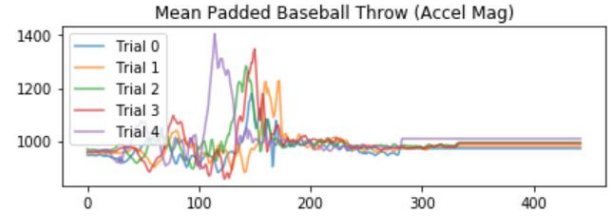
1. **Visualize the data.** Look for patterns. Brainstorm how to leverage those patterns in your algorithms

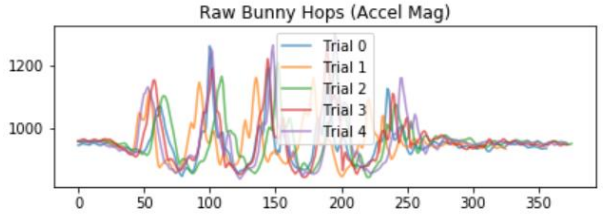
2. **Make all signals across all trials the same length** to simplify comparison. Do this via numpy's padding functionality.

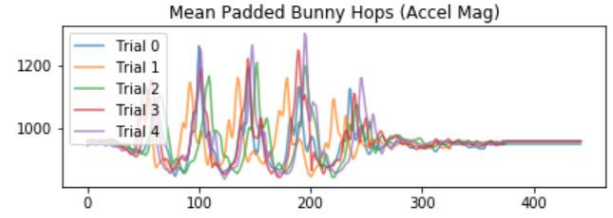


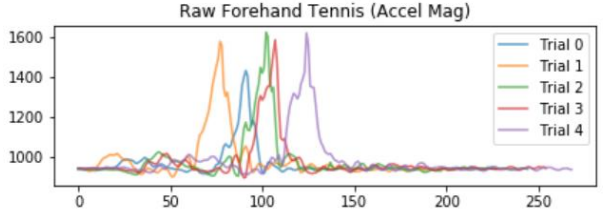


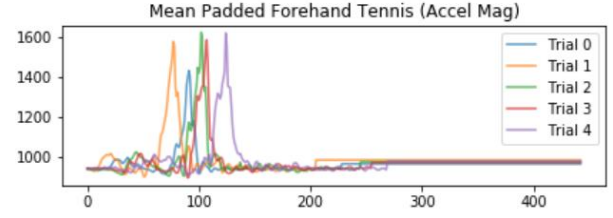


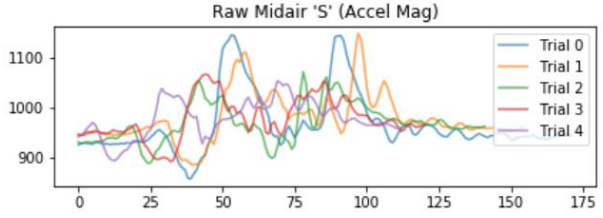


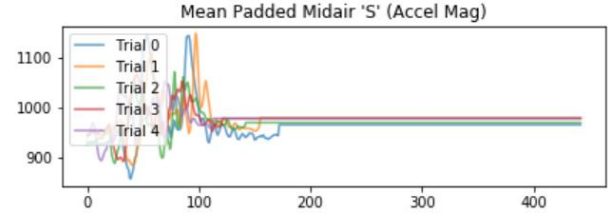


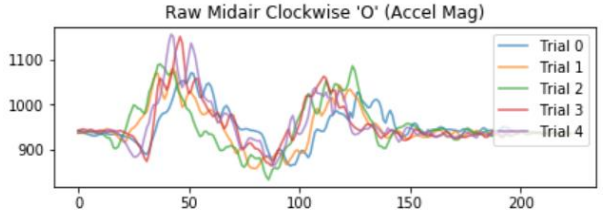


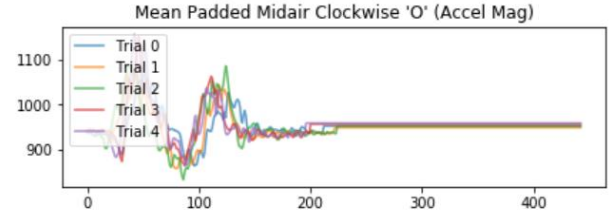












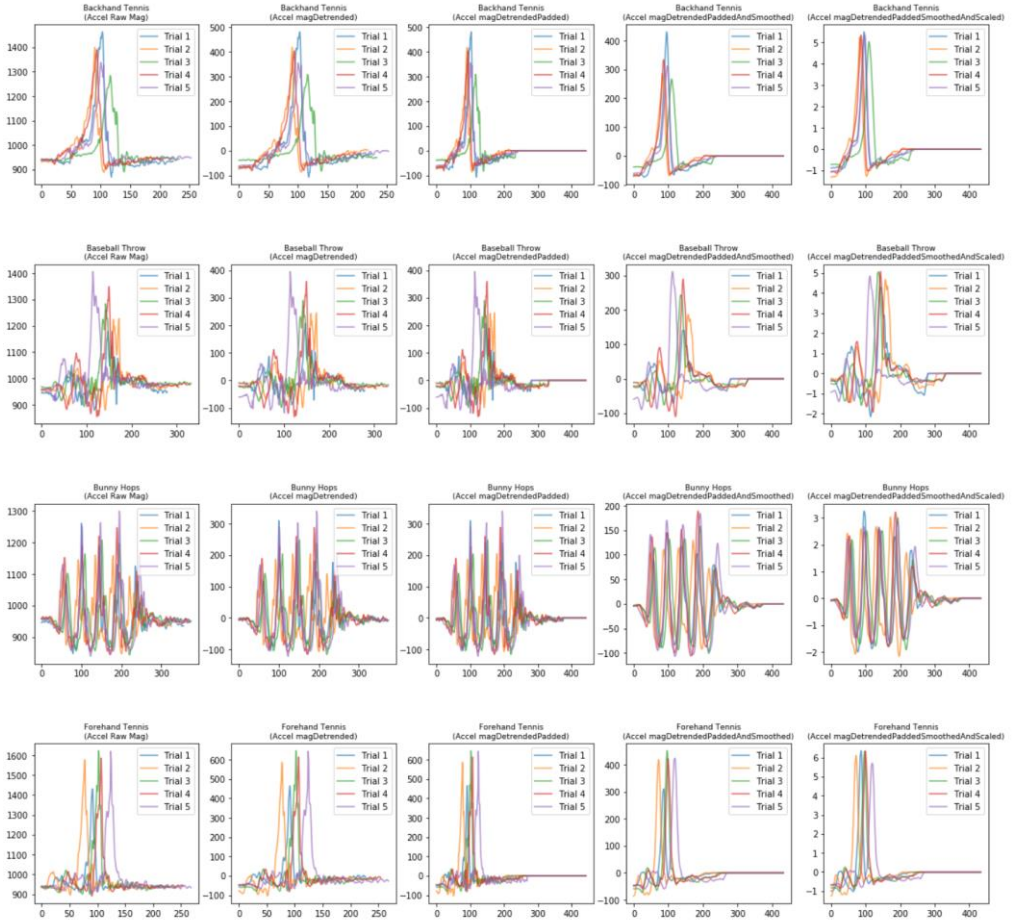
A3: SHAPE-BASED GESTURE RECOGNITION

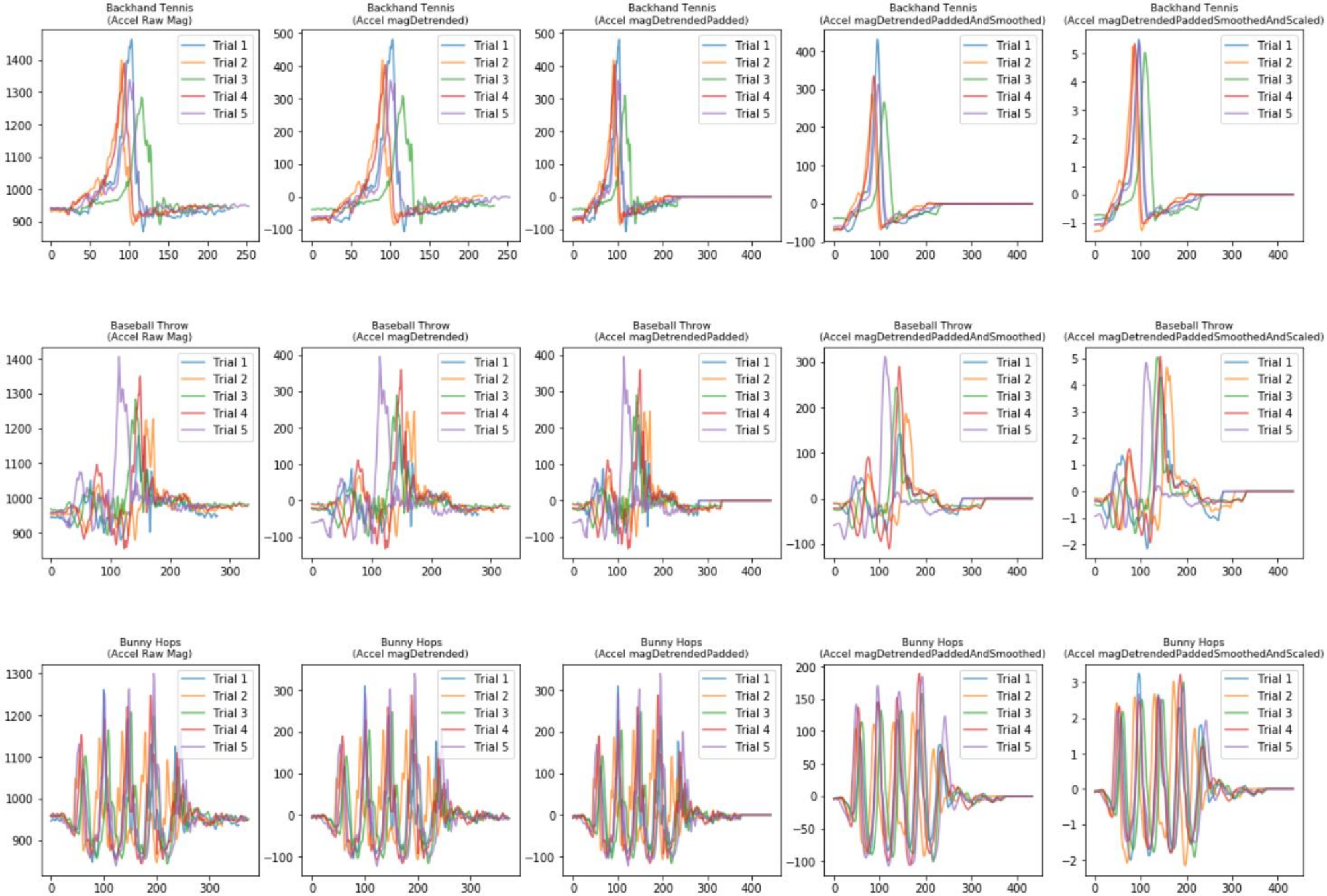
GENERAL APPROACH

1. **Visualize the data.** Look for patterns. Brainstorm how to leverage those patterns in your algorithms

2. **Make all signals across all trials the same length** to simplify comparison. Do this via numpy's padding functionality.

3. **Play around with various transforms of the data.** Smoothed. Detrended. Scaled?




Backhand Tennis
(Accel Raw Mag)
Backhand Tennis
(Accel magDetrended)
Backhand Tennis
(Accel magDetrendedPadded)
Backhand Tennis
(Accel magDetrendedPaddedAndSmoothed)
Backhand Tennis
(Accel magDetrendedPaddedSmoothedAndScaled)
Baseball Throw
(Accel Raw Mag)
Baseball Throw
(Accel magDetrended)
Baseball Throw
(Accel magDetrendedPadded)
Baseball Throw
(Accel magDetrendedPaddedAndSmoothed)
Baseball Throw
(Accel magDetrendedPaddedSmoothedAndScaled)
Bunny Hops
(Accel Raw Mag)
Bunny Hops
(Accel magDetrended)
Bunny Hops
(Accel magDetrendedPadded)
Bunny Hops
(Accel magDetrendedPaddedAndSmoothed)
Bunny Hops
(Accel magDetrendedPaddedSmoothedAndScaled)
Trial 1
Trial 2
Trial 3
Trial 4
Trial 5

A3: SHAPE-BASED GESTURE RECOGNITION

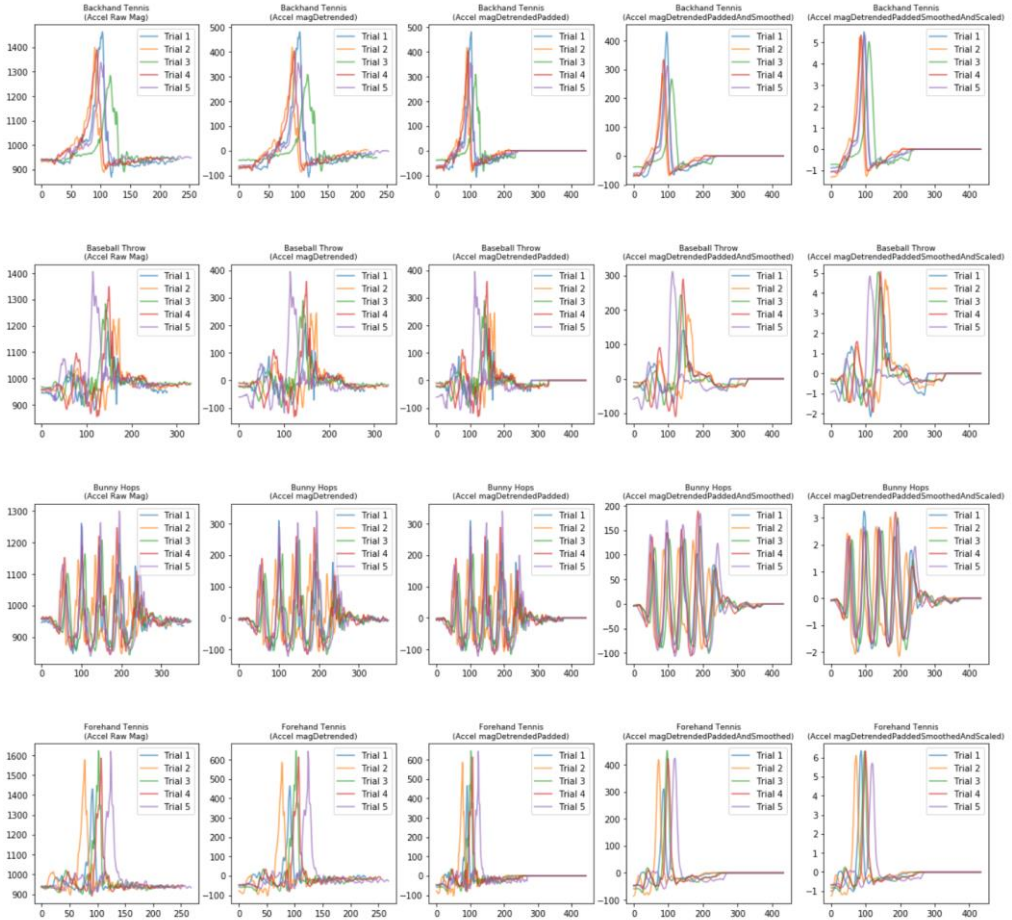
GENERAL APPROACH

1. **Visualize the data.** Look for patterns. Brainstorm how to leverage those patterns in your algorithms

2. **Make all signals across all trials the same length** to simplify comparison. Do this via numpy's padding functionality.

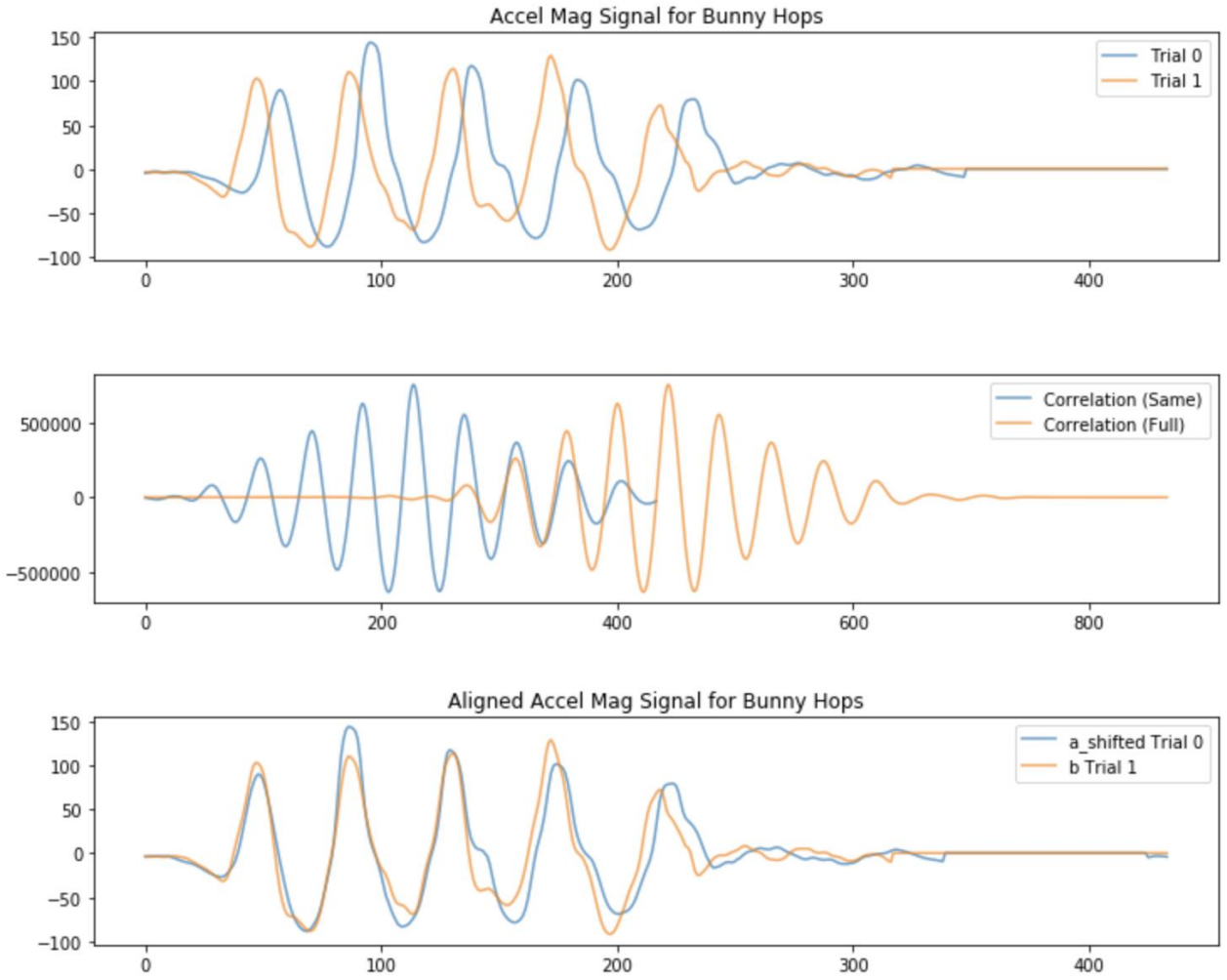
3. **Play around with various transforms of the data.** Smoothed. Detrended. Scaled?

4. **Determine strategy/algorithm for comparing two signals.** Euclidean distance? Dynamic time warping?



HINT: EXPLORE CROSS CORRELATION TO ALIGN TWO SIGNALS BEFORE COMPARISON

See:
https://www.mathworks.com/help/signal/examples/measuring-signal-similarities.html



A3: SHAPE-BASED GESTURE RECOGNITION

GENERAL APPROACH

1. **Visualize the data.** Look for patterns. Brainstorm how to leverage those patterns in your algorithms

2. **Make all signals across all trials the same length** to simplify comparison. Do this via numpy's padding functionality.

3. **Play around with various transforms of the data.** Smoothed. Detrended. Scaled?

4. **Determine strategy/algorithm for comparing two signals.** Euclidean distance? Dynamic time warping?

5. **Setup experiment infrastructure to help you evaluate and compare performance.** In this case, we are using k-fold cross validation.

A3: SHAPE-BASED GESTURE RECOGNITION

GENERAL APPROACH

1. **Visualize the data.** Look for patterns. Brainstorm how to leverage those patterns in your algorithms

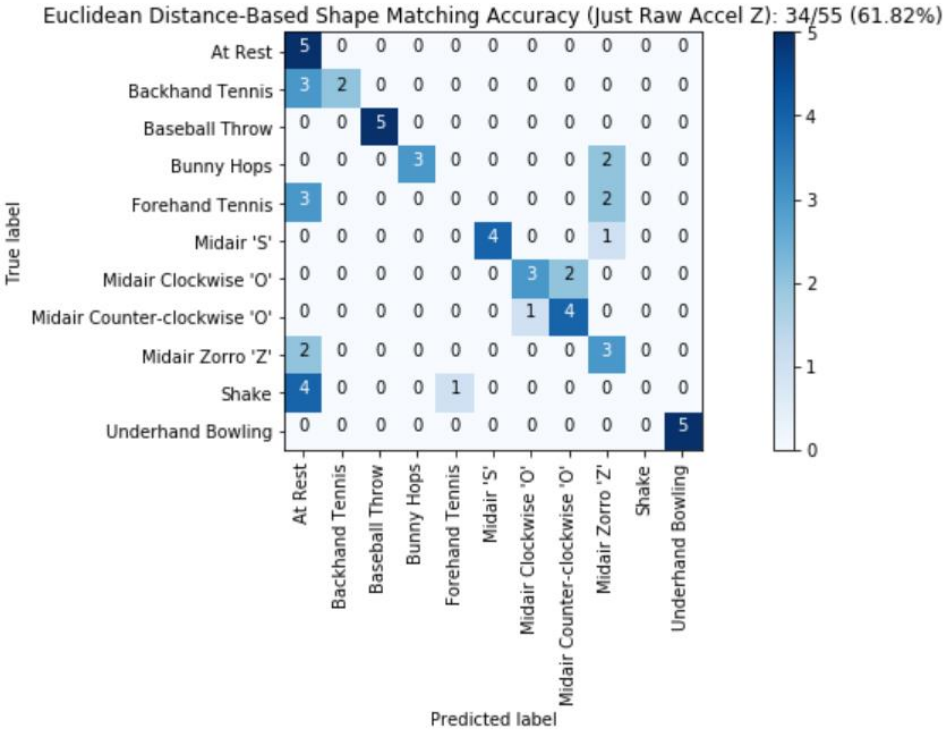
2. **Make all signals across all trials the same length** to simplify comparison. Do this via numpy's padding functionality.

3. **Play around with various transforms of the data.** Smoothed. Detrended. Scaled?

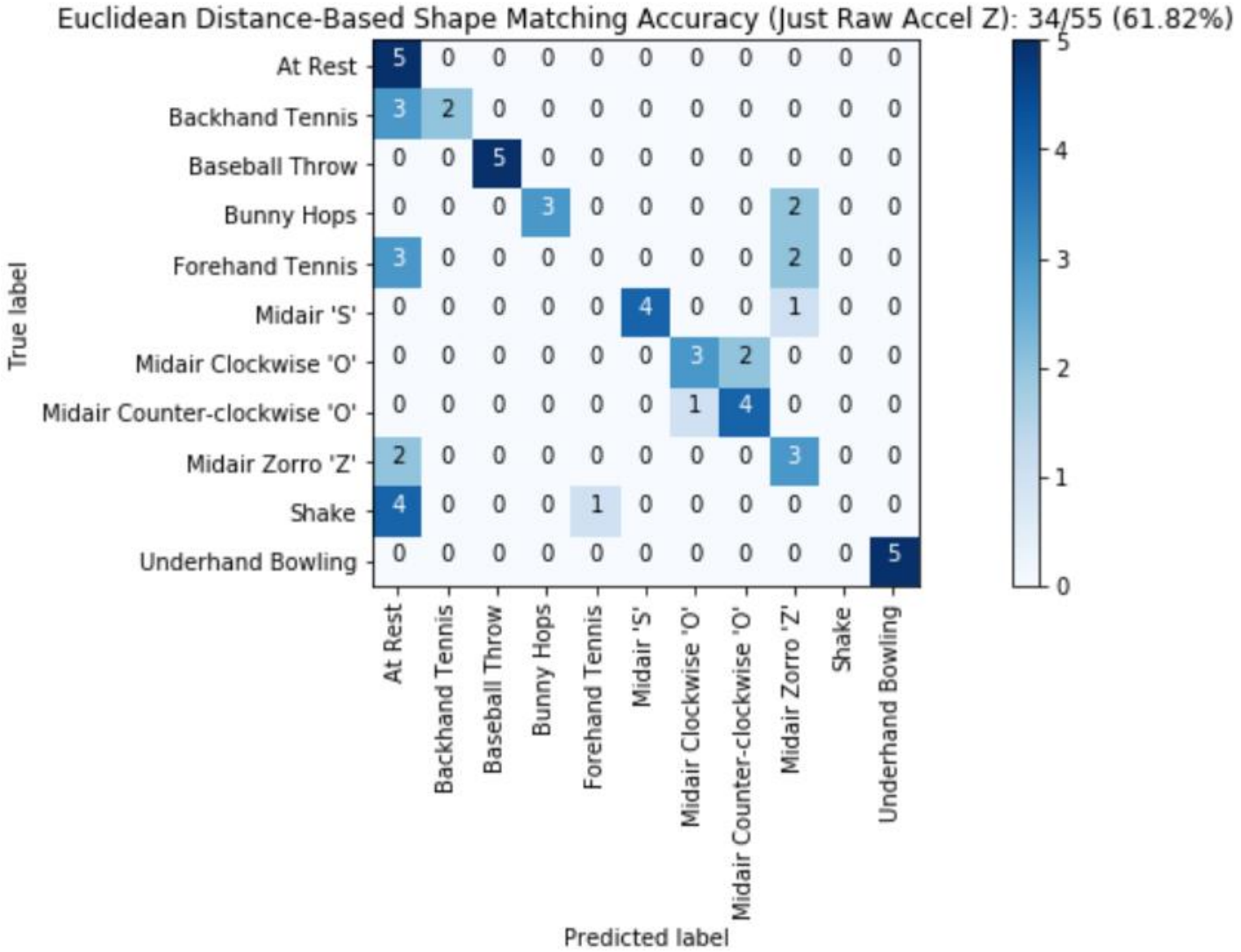
4. **Determine strategy/algorithm for comparing two signals.** Euclidean distance? Dynamic time warping?

5. **Setup experiment infrastructure to help you evaluate and compare performance.** In this case, we are using k-fold cross validation.

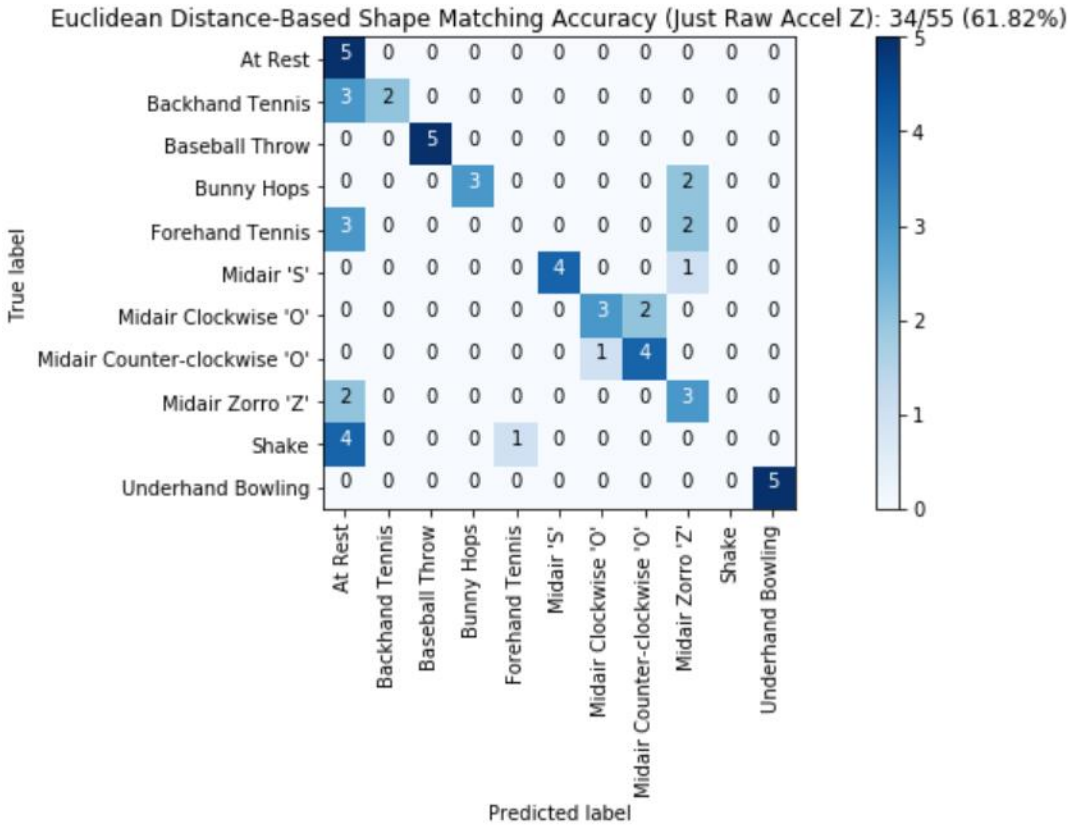
6. **Create visualizations of your results to help inform your algorithm.** Iterate. Iterate.

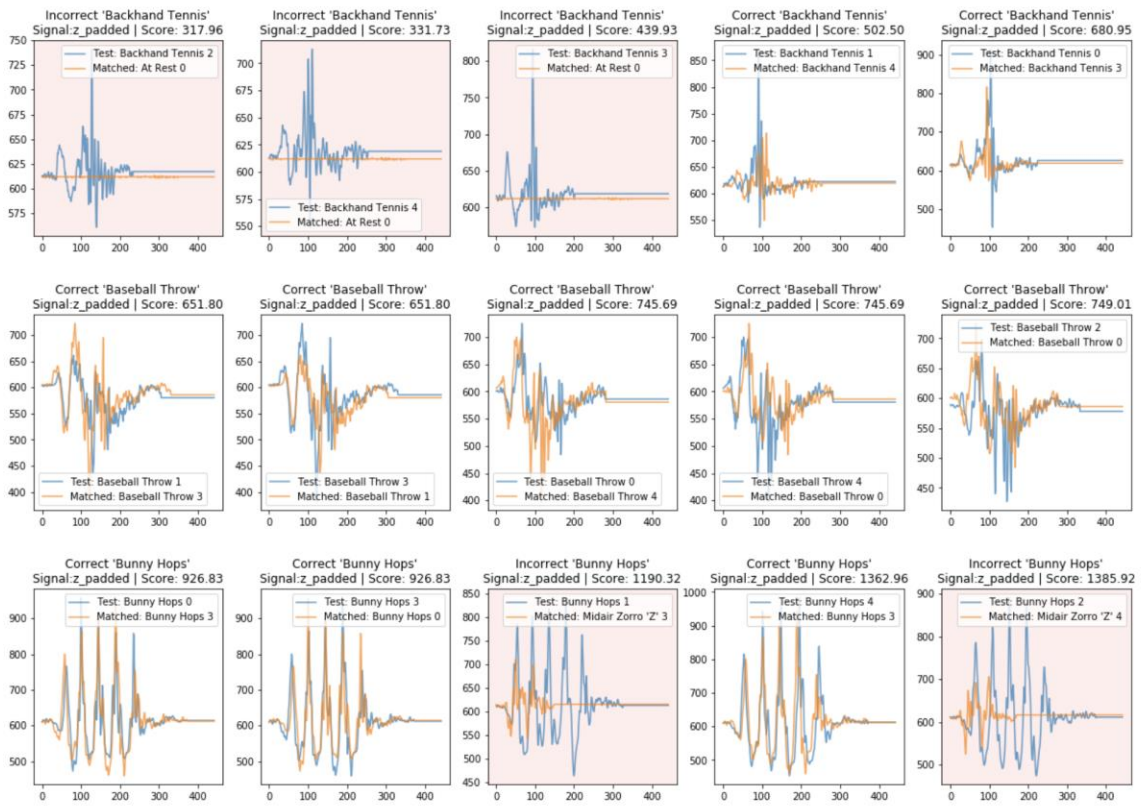


VISUALIZATIONS: CONFUSION MATRIX



VISUALIZE CORRECT & INCORRECT MATCHES

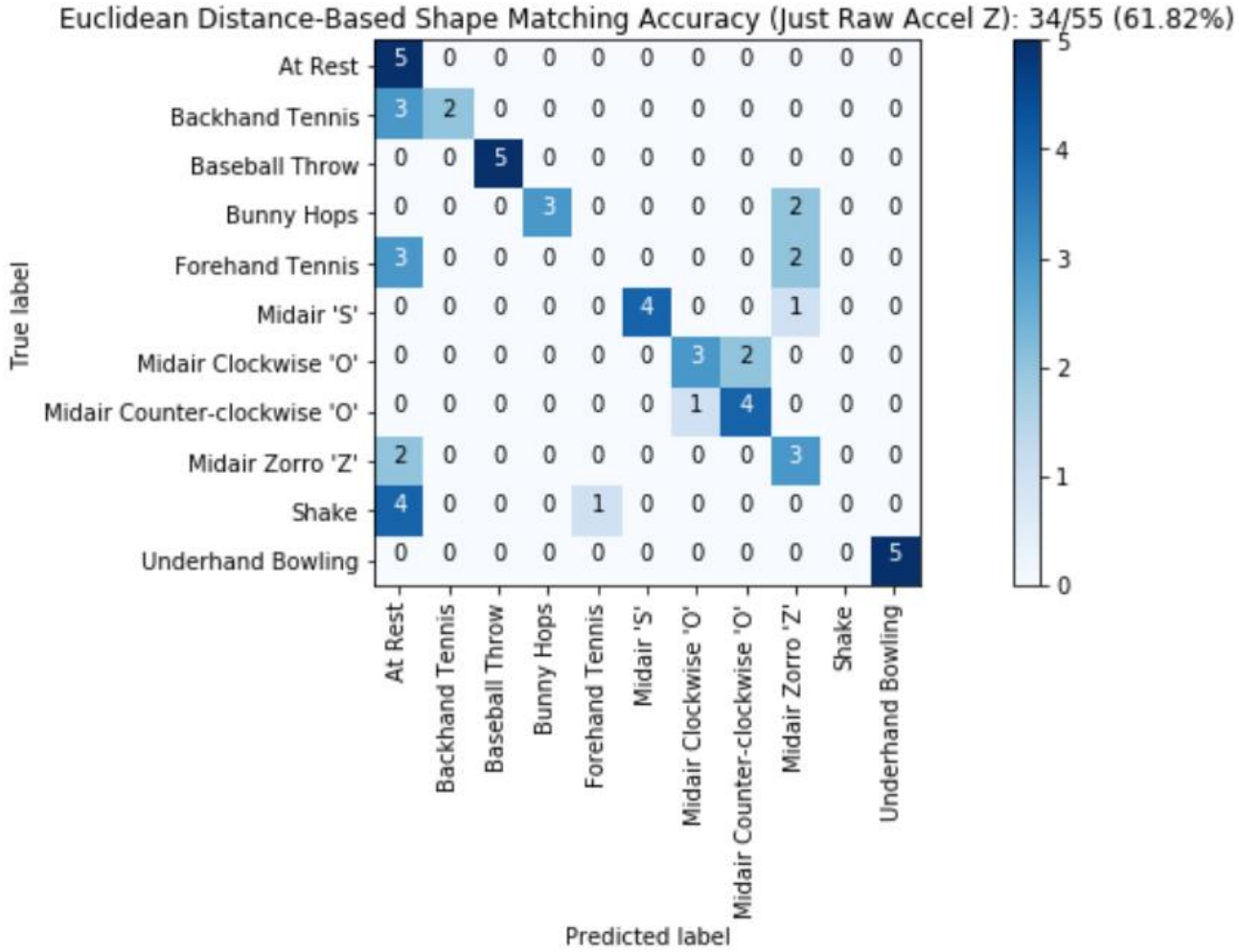


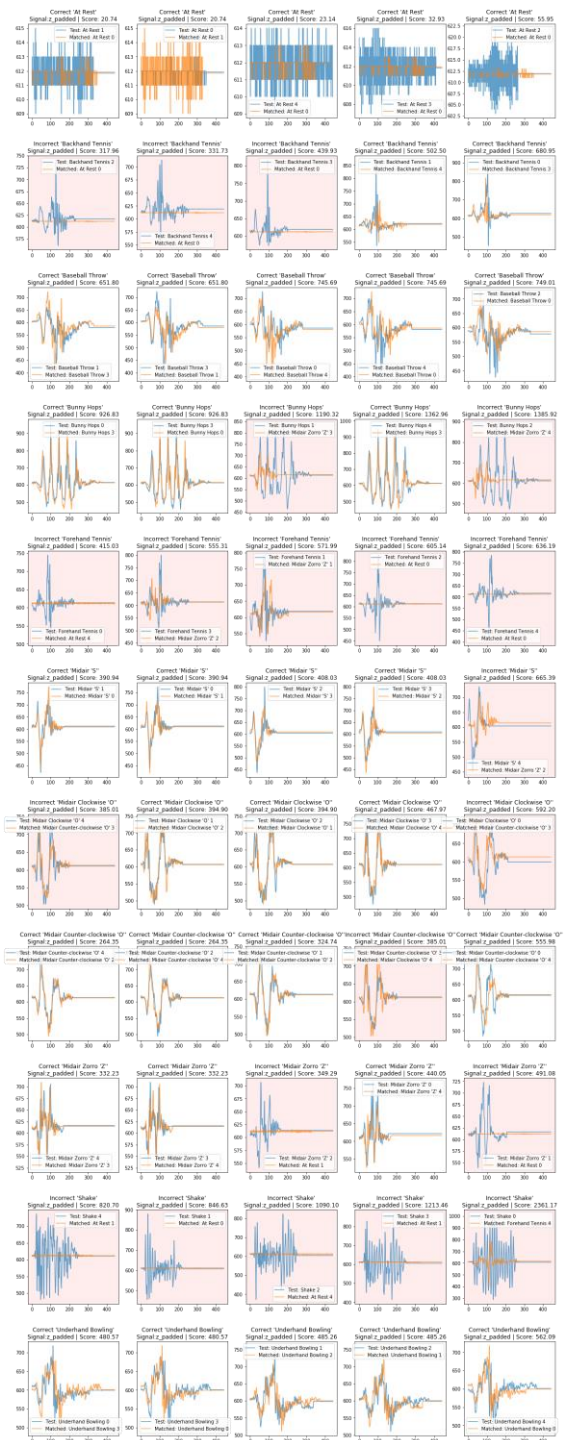


Each row is a gesture. Each column is the best scoring match between the test gesture and the fold gestures. The cells with a “light red” background are the incorrect matches. The cells with a “white background” are the correct matches.

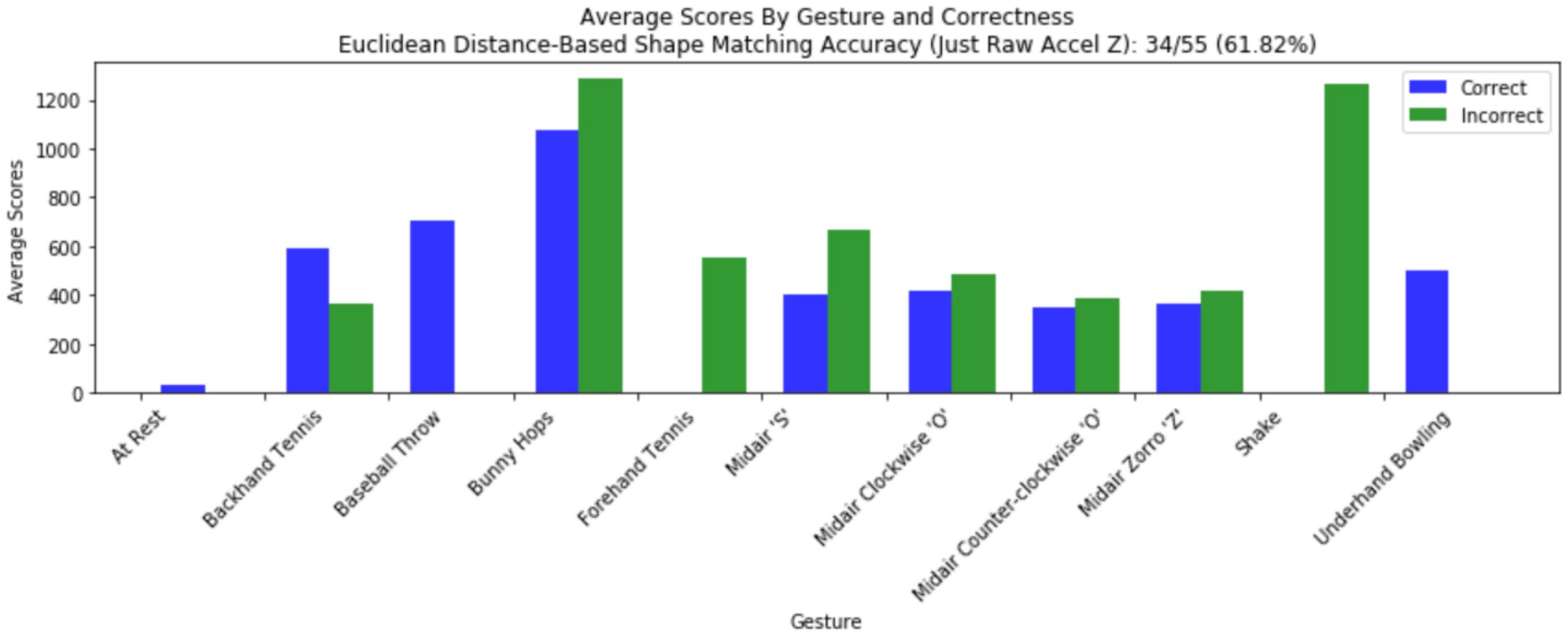
A3: SHAPE-BASED GESTURE RECOGNITION

VISUALIZING ALL BEST MATCH RESULTS

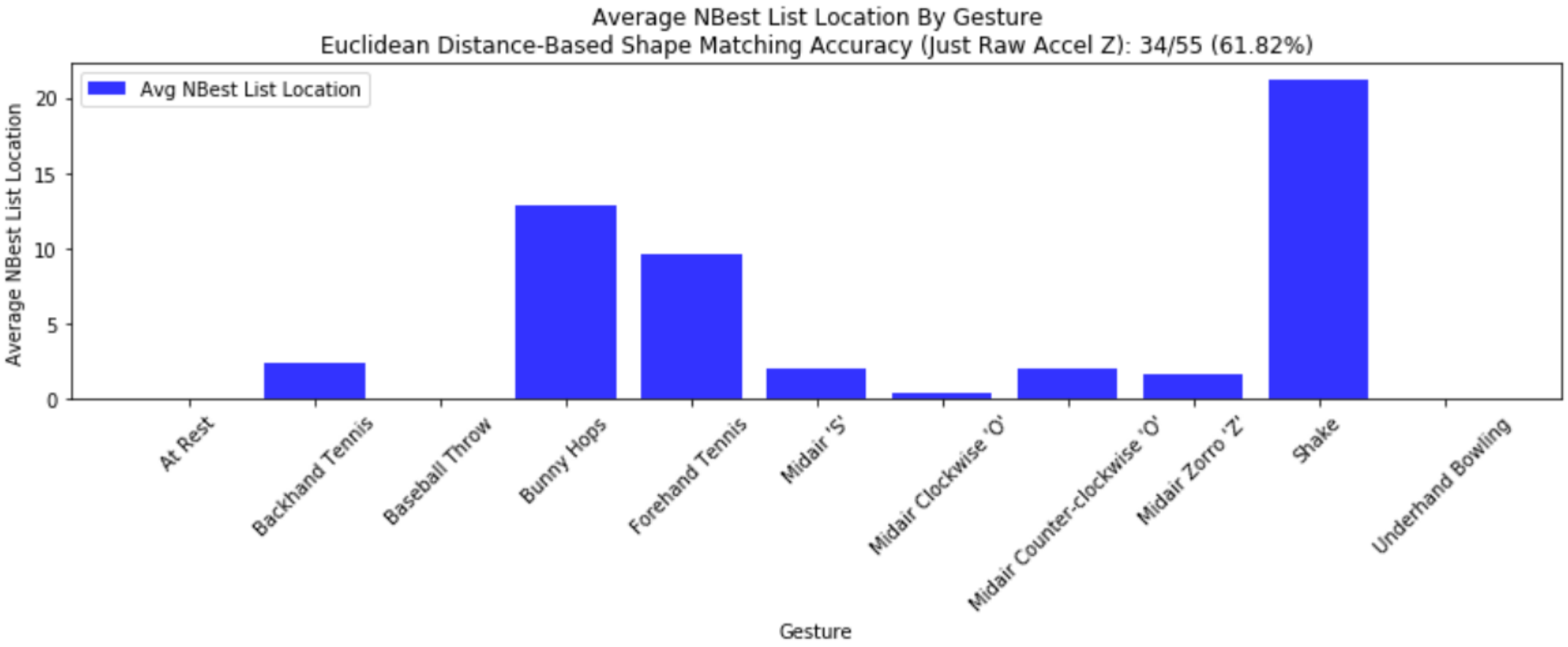




VISUALIZATIONS: AVERAGE SCORES

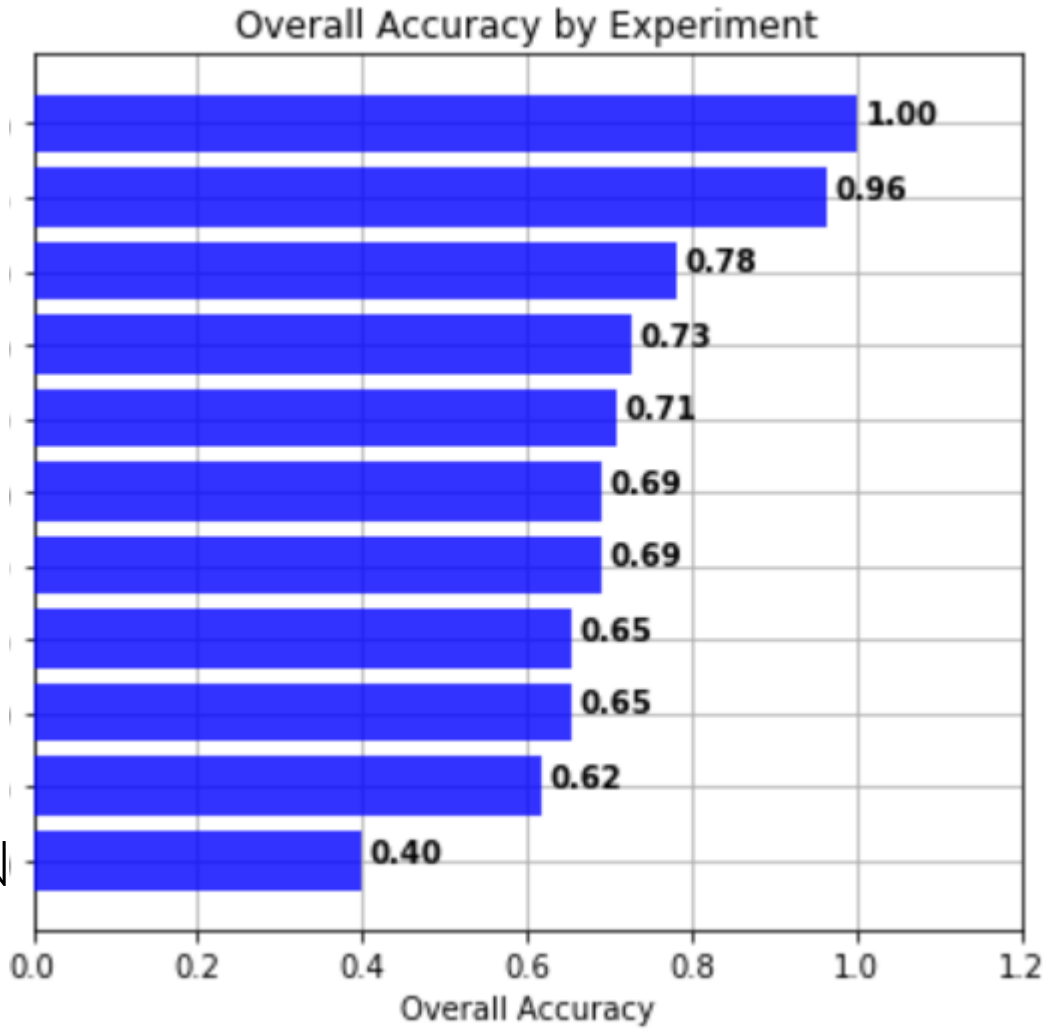


VISUALIZATIONS: AVERAGE NBEST LIST LOCATION



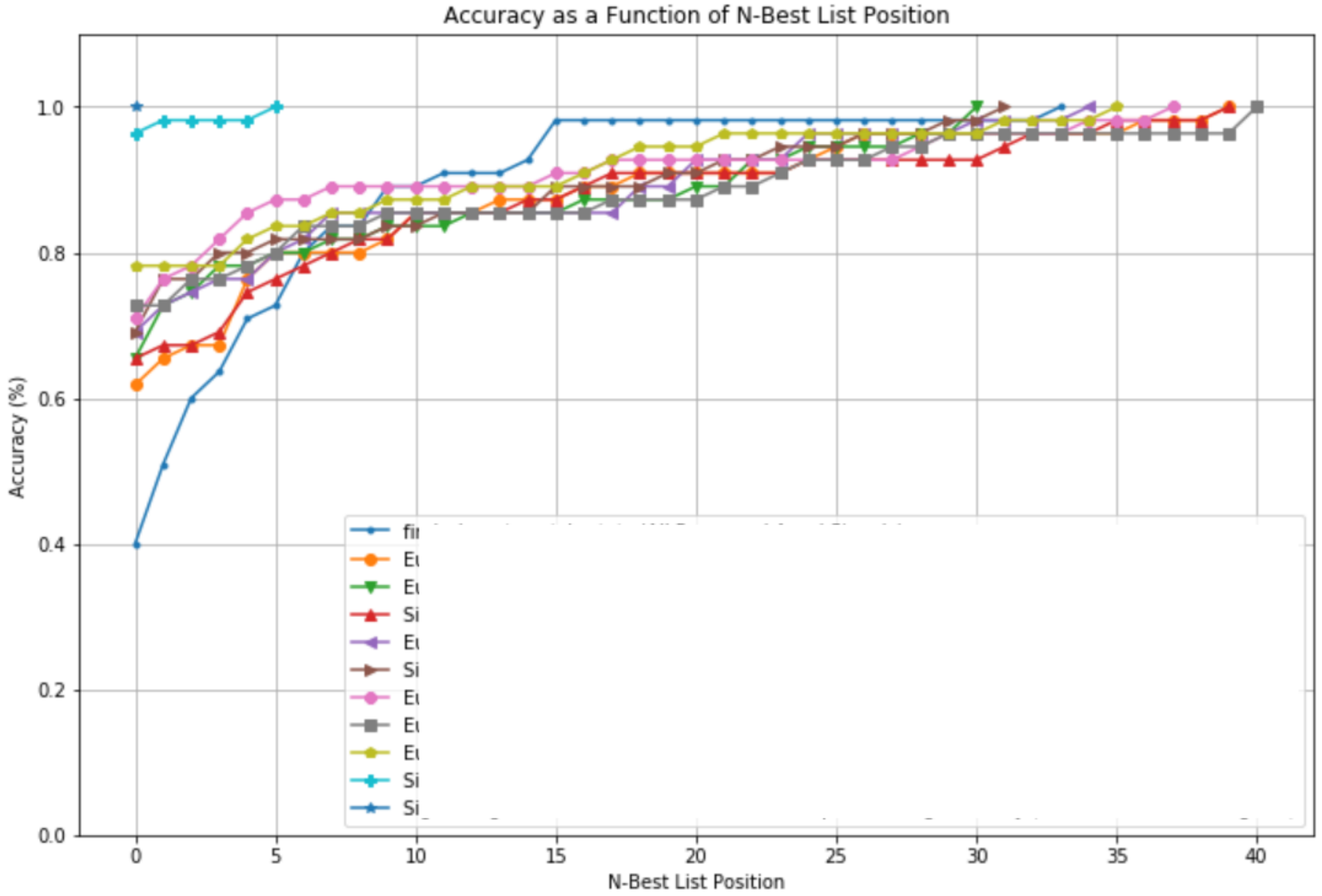
VISUALIZATIONS: COMPARING ALGORITHMS/APPROACHES

Algorithm 1



Algorithm N

VISUALIZATIONS: COMPARING ALGORITHMS/APPROACHES



Project Jupyter | Home

jupyter.org

jupyter

Install About Us Community Documentation NBViewer Widgets Blog

Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.

JUPYTER NOTEBOOK

TUTORIAL NOTEBOOKS

jonfroehlich / CSE599Sp2019

Unwatch 7 | Star 4 | Fork 0

Code | Issues 1 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

Branch: master | CSE599Sp2019 / Notebooks / | Create new file | Upload files | Find file | History

jonfroehlich updated comment — Latest commit 6347c62 2 hours ago

..		
MatplotlibPyplotTutorial.ipynb	added figure to matplotlib tutorial	16 hours ago
NumpyTutorial.ipynb	Added in more dynamic invocation examples	2 hours ago
PythonTutorial.ipynb	updated comment	2 hours ago

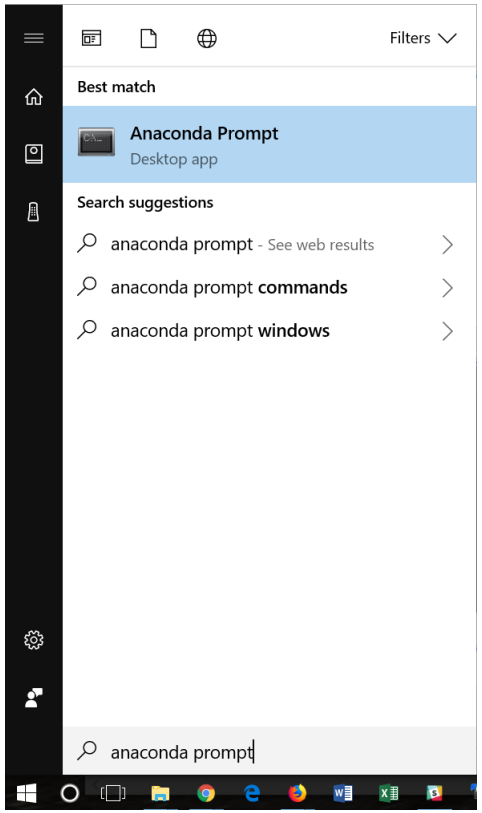
https://github.com/jonfroehlich/CSE599Au2019/tree/master/Notebooks

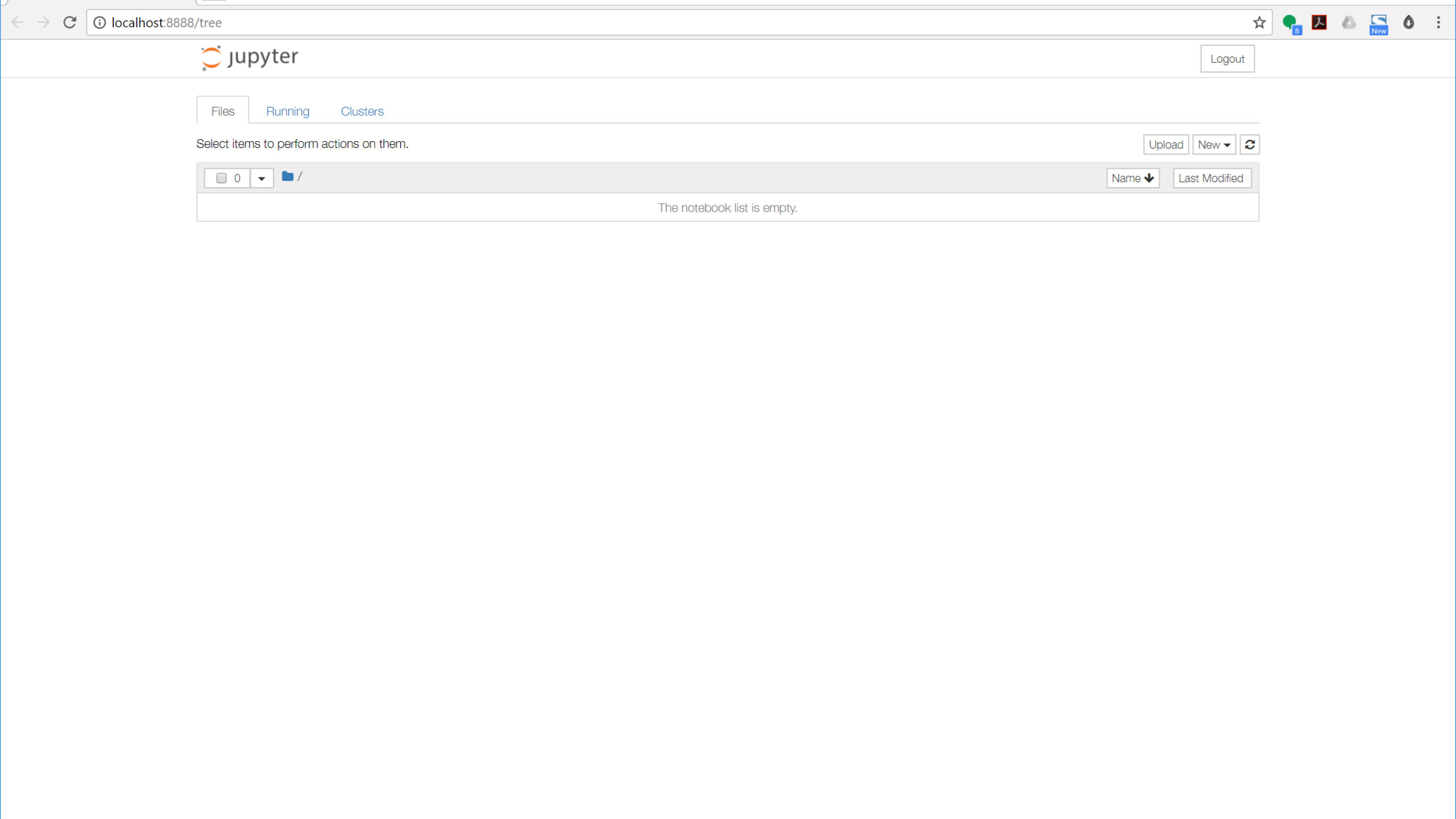
STARTING A FRESH JUPYTER NOTEBOOK

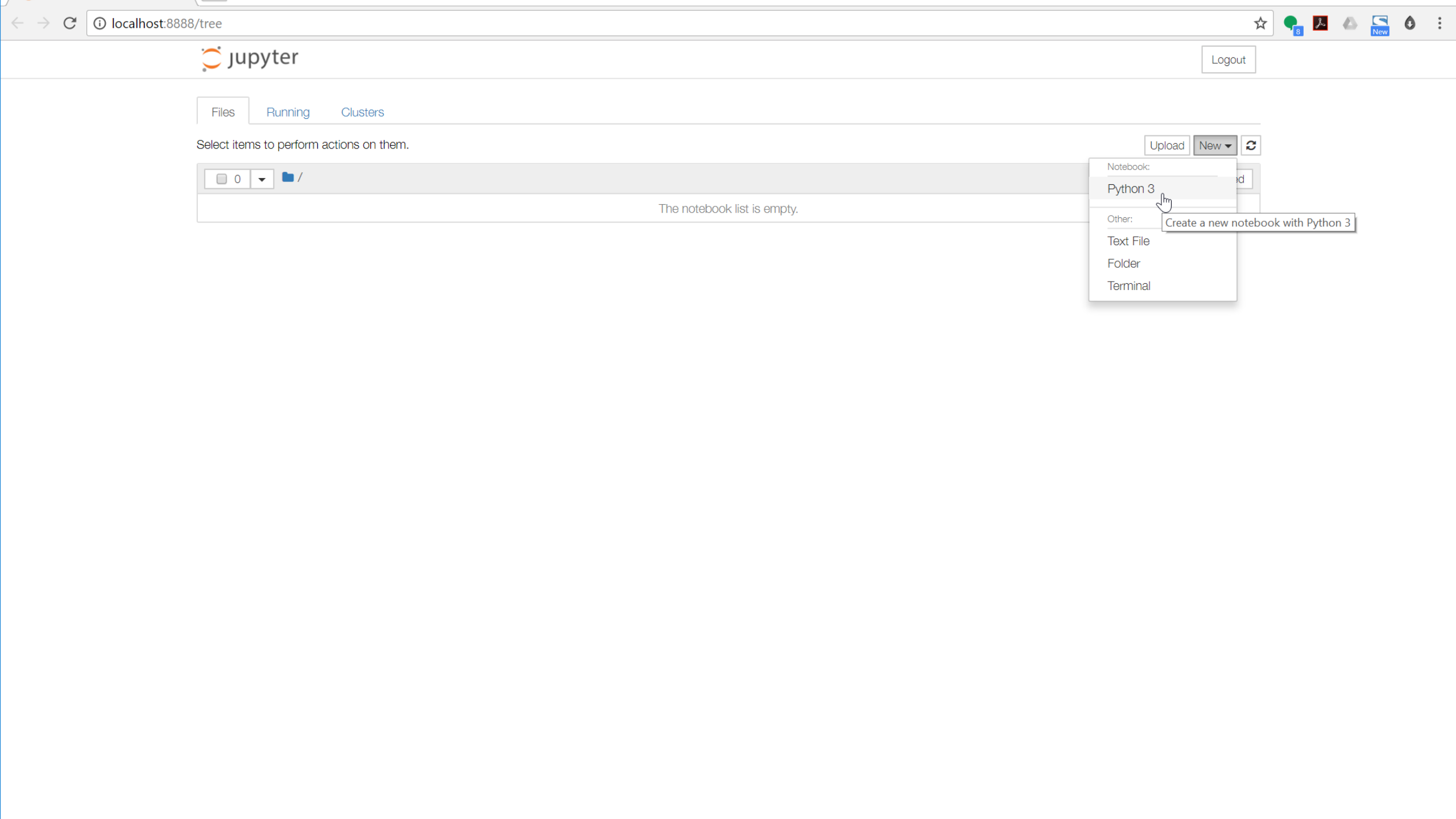
On a Mac

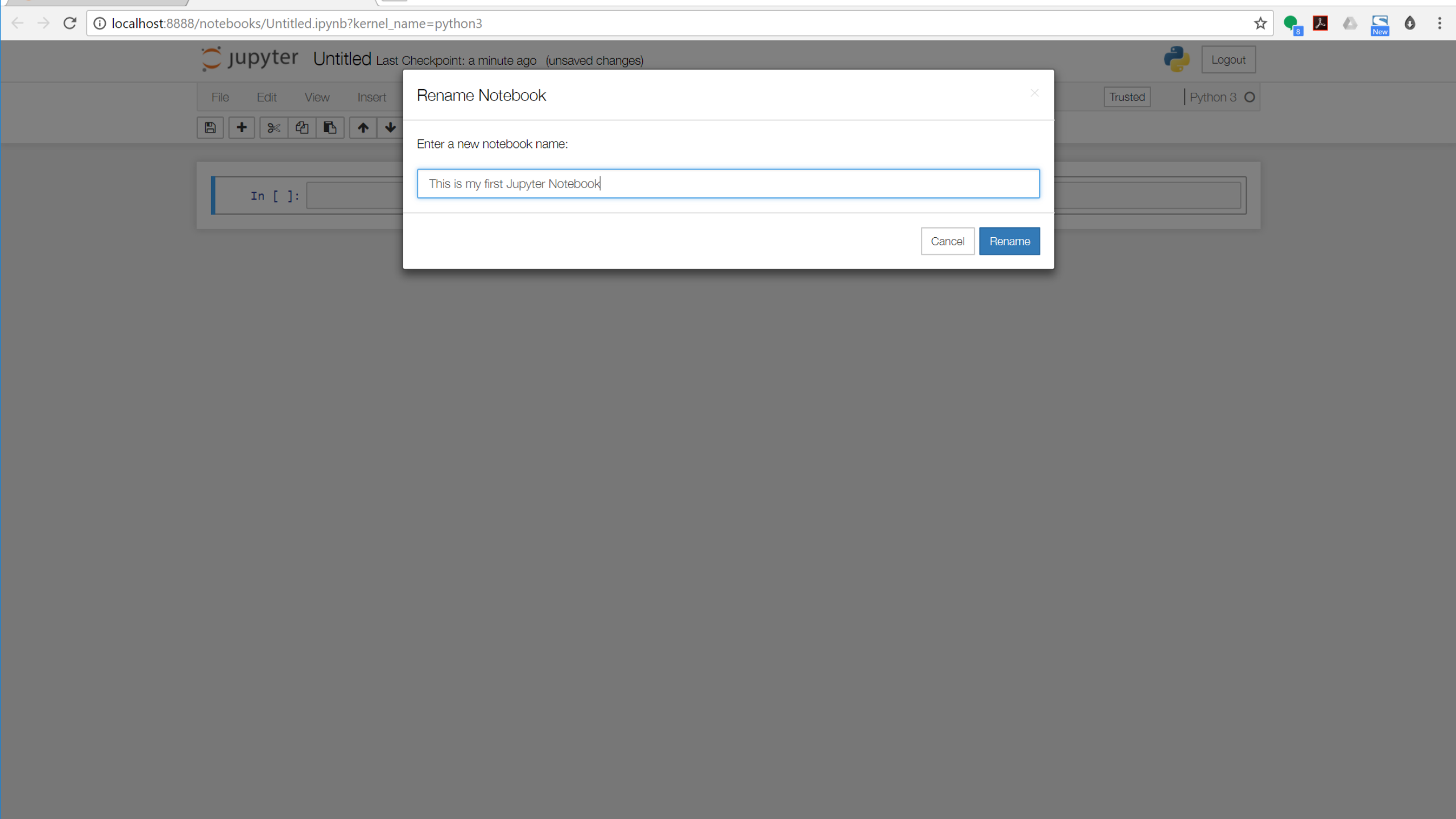
```
> mkdir funtimes
> cd funtimes
> jupyter notebook
```

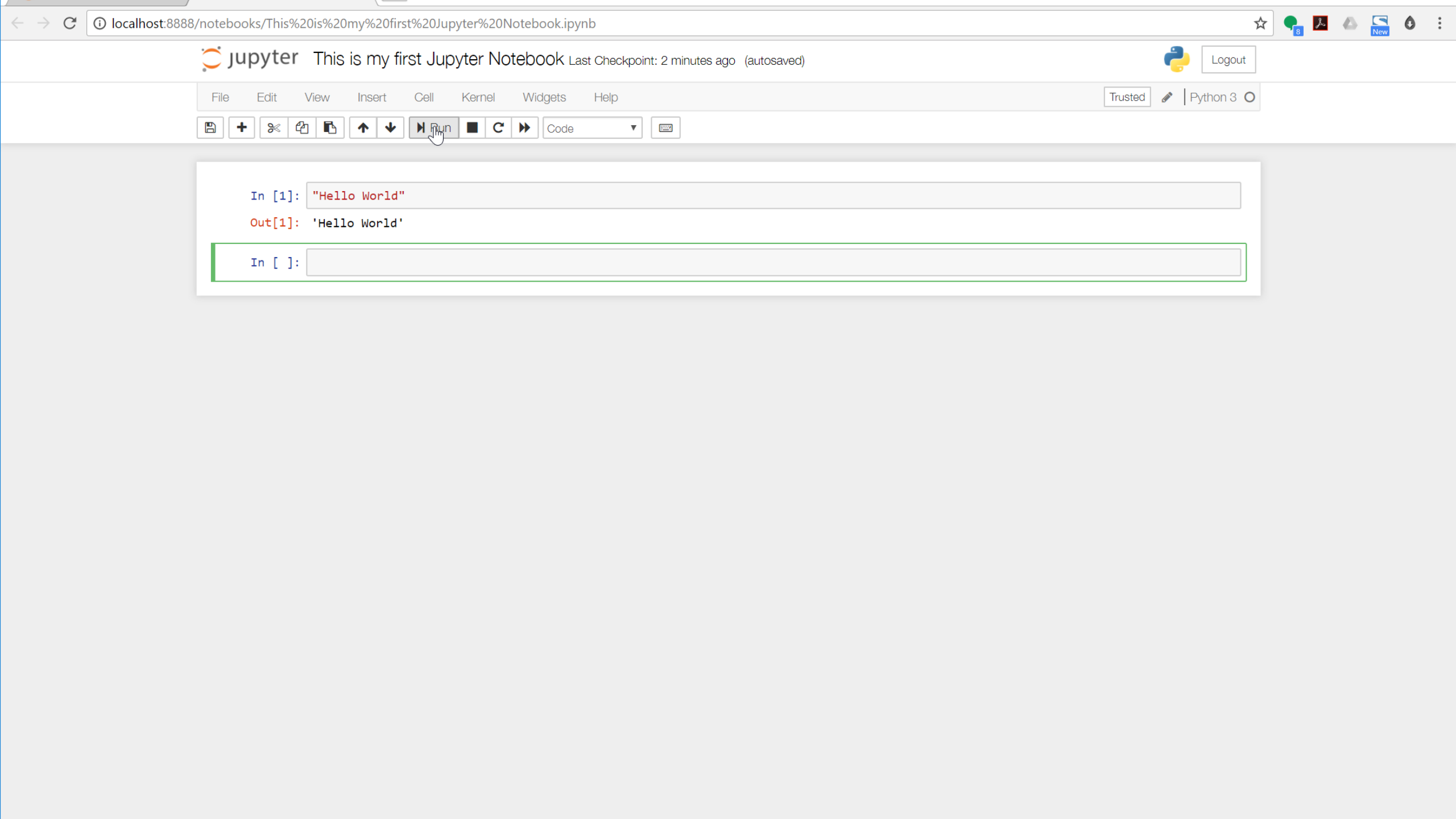
On Windows




localhost:8888/tree
jupyter
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
0
/
Name
Last Modified
The notebook list is empty.


localhost:8888/tree
jupyter
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
0
/
The notebook list is empty.
Notebook:
Python 3
Other:
Text File
Folder
Terminal
Create a new notebook with Python 3


localhost:8888/notebooks/Untitled.ipynb?kernel_name=python3
jupyter Untitled Last Checkpoint: a minute ago (unsaved changes)
Logout
File Edit View Insert
Trusted
Python 3
In [ ]:
Rename Notebook
Enter a new notebook name:
This is my first Jupyter Notebook
Cancel
Rename


localhost:8888/notebooks/This%20is%20my%20first%20Jupyter%20Notebook.ipynb
jupyter This is my first Jupyter Notebook Last Checkpoint: 2 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [1]: "Hello World"
Out[1]: 'Hello World'
In [ ]:

In [1]:
```
"Hello World"
```

Out[1]: 'Hello World'

In [3]:
```
myList = [1, 2, 3, 4, 5]
print(myList)
```

```
[1, 2, 3, 4, 5]
```

In [ ]:

localhost:8888/notebooks/This%20is%20my%20first%20Jupyter%20Notebook.ipynb

jupyter This is my first Jupyter Notebook Last Checkpoint: 8 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [1]:
```
"Hello World"
```

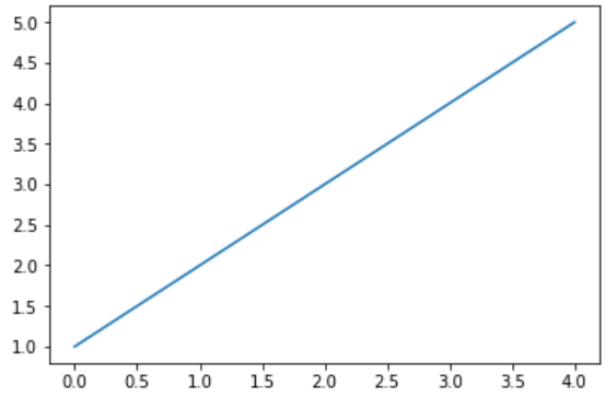
Out[1]:
```
'Hello World'
```

In [3]:
```
myList = [1, 2, 3, 4, 5]
print(myList)
```

```
[1, 2, 3, 4, 5]
```

In [6]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.plot(myList)
```

Out[6]:
```
[<matplotlib.lines.Line2D at 0x1e822d0e8d0>]
```

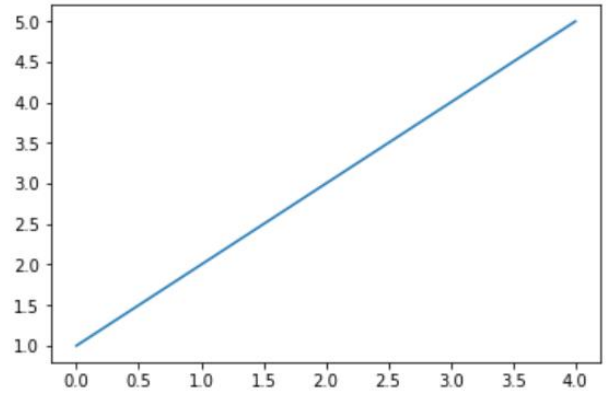


In [ ]:

```
fig, ax = plt.subplots()
ax.plot(myList)
```

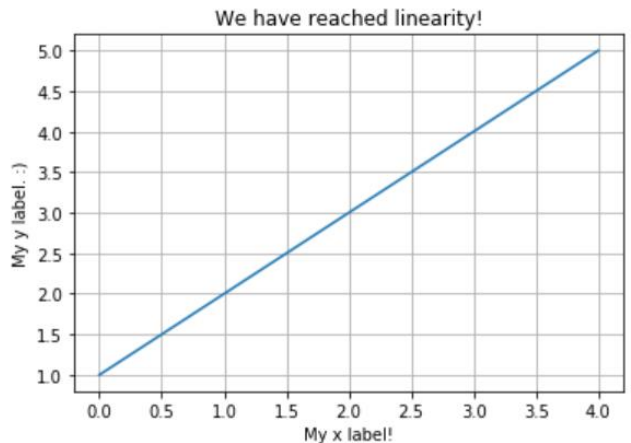
Out[6]:

```
[<matplotlib.lines.Line2D at 0x1e822d0e8d0>]
```



In [8]:

```
fig, ax = plt.subplots()
ax.plot(myList)
ax.set(xlabel='My x label!', ylabel='My y label. :)',
       title='We have reached linearity!')
ax.grid()
```

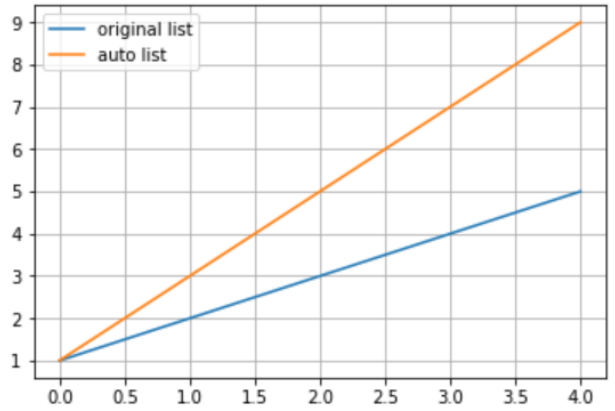


```
# range takes a start val, end val, and step
myList2 = list(range(1, 10, 2))
print(myList2)
```

```
[1, 3, 5, 7, 9]
```

In [16]:

```
# Let's plot both of these lists
fig, ax = plt.subplots()
ax.plot(myList, label="original list")
ax.plot(myList2, label="auto list")
ax.legend()
ax.grid()
```



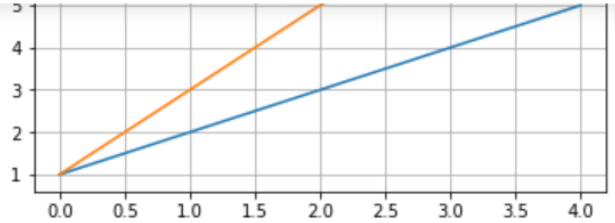
In [19]:

```
# We can also have HashMaps or dictionaries--Python calls them dicts
mapNamesToAges = {"Camden" : 1.5, "Gavin" : 3.3, "Susan": 4.5, "Billy": 4.4, "Frankie" : 6}
print(mapNamesToAges)
```

```
{'Camden': 1.5, 'Gavin': 3.3, 'Susan': 4.5, 'Billy': 4.4, 'Frankie': 6}
```

In [20]:

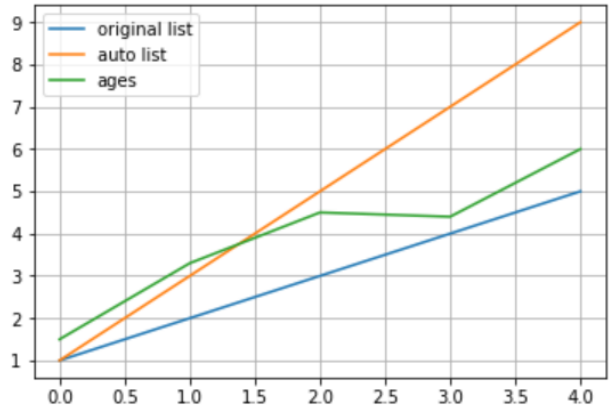
```
# Let's plot them all!
fig, ax = plt.subplots()
ax.plot(myList, label="original list")
ax.plot(myList2, label="auto list")
ax.plot(mapNamesToAges.values(), label="ages")
ax.legend()
ax.grid()
```



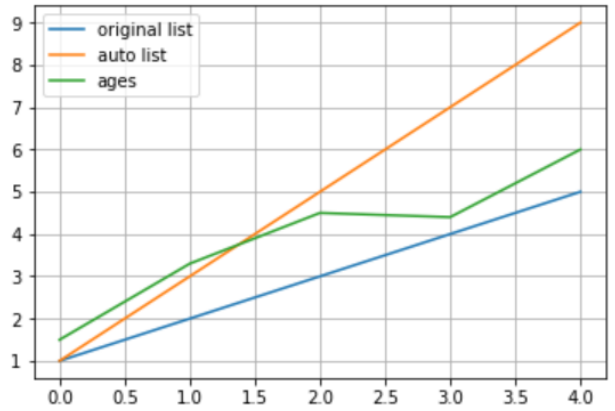
In [19]:
```
# We can also have HashMaps or dictionaries--Python calls them dicts
mapNamesToAges = {"Camden" : 1.5, "Gavin" : 3.3, "Susan": 4.5, "Billy": 4.4, "Frankie" : 6}
print(mapNamesToAges)
```

```
{'Camden': 1.5, 'Gavin': 3.3, 'Susan': 4.5, 'Billy': 4.4, 'Frankie': 6}
```

In [20]:
```
# Let's plot them all!
fig, ax = plt.subplots()
ax.plot(myList, label="original list")
ax.plot(myList2, label="auto list")
ax.plot(mapNamesToAges.values(), label="ages")
ax.legend()
ax.grid()
```



Markdown Support!



Markdown Support!

Jupyter Notebook also supports markdown! From the dropdown menu, select Cell->Cell Type->Markdown

```
# Markdown Support!
Jupyter Notebook also supports markdown! From the dropdown menu, select Cell->Cell Type->Markdown
```

JUPYTER NOTEBOOK

NOTEBOOK SUPPORTS MARKDOWN

1

I am a new to Jupyter Notebook too, must say that their documentation is awful as they say what we **can do** instead of **HOW** to do it. I have opened already made '.ipynb' notebook and figured it out, I will list a couple of them from **unexplained documentation**:

1. Make text ITALIC: `*Italic*`
2. Make text BOLD: `**Bold**`
3. List item as a bullet: dash and space `-`
4. List item as a number: Simple as number and dot `1.`
5. Indenting text: Greater than and space `>`
6. Inline code span: Back quotation mark " `` ` "
7. Block of code: Triple back quotation marks " ```` ``` "
8. Link a section: `[Title of Section](#title-of-section)`
9. Hyperlink: `[Text](URL)`

I hope everyone has found their solution.

share improve this answer

answered Apr 26 at 22:48

Uki.

11 • 1

add a comment

0

1

0

Hot I

Wo for

Ho

Co

Ca "m

An

Ho fro

Ho

Wh inp