SIGNAL PROCESSING IN UBICOMP

CSE 590 Ubiquitous Computing | Lecture 2 | April 5

Jon Froehlich • Liang He (TA)

DESIGN USE BUILD

MAKEABILITY LAB

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY of WASHINGTON

How are things so far?

ABOUT THIS CLASS

PEDAGOGICAL APPROACH

Edit

Updated Github Repo

Apr 2 at 2:06pm

Jon Froehlich

All Sections

In this class, you will learn primarily by doing (i.e., *constructionism*). The lectures will be used to give you a basic entrée into a given topic but the assignments will really allow you to dive deeper and learn the material.

While we will use Android, this is not an Android class. While we will use Arduino, this is not an Arduino class. While we will use OpenCV, this is not a computer vision class. Etc. Etc. This class represents the highly disparate and diverse Ubiquitous Computing! :)

So, I will try to keep posting code and tutorials that may help you ramp up to the various technologies we use in class to help you sidestep some of the learning curves inherent in learning any new dev environment or technology.

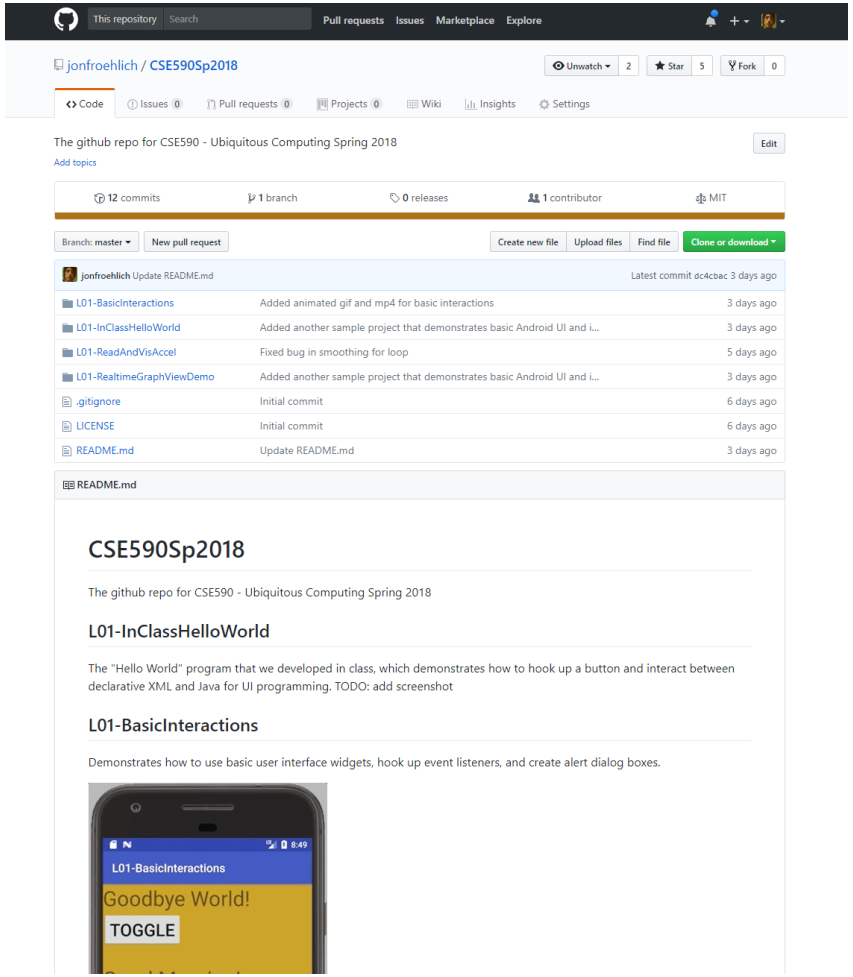
Please check out https://github.com/jonfroehlich/CSE590Sp2018 . Let me know if any of the code samples don't run or if there are problems. Also, if you *are* an Android expert already, feel free to push me on some of my implementations. I am trying to keep things simple and go with conventional implementation approaches... but I may have occasionally erred.

Don't forget, the mid-point checkin for A1, the readings, and the background survey are all due this week!

Jon

Search entries or author | Unread

Reply



https://github.com/jonfroehlich/CSE590Sp2018

Spring 2018

Home
Announcements
Assignments
Discussions
Grades
People
Pages
Files
Syllabus
Outcomes
Quizzes
Modules
Conferences
Collaborations
Chat
Attendance
UW Libraries
Add 4.0 Grade Scale
Panopto Recordings
Settings

A1: Step Tracker

Published | Edit

Goal

In this assignment, you will build your own step tracker using your Android smartphone's built-in accelerometer (and gyro, optional) along with basic signal processing algorithms and heuristics. You should **not** use machine learning for this assignment (*e.g.*, supervised learning).

There will be a midpoint check-in on April 5 (link). You will demonstrate a working version of your prototype to the TA, Liang He, during classtime on April 12.

Learning Goals

- Introduce and learn the basics of Android development (*e.g.*, Android Studio, using the emulator, deploying to a device, debugging, the Android architecture).
- Introduce and learn basic methods for accessing and processing built-in sensors on Android.
- Introduce and learn how to apply basic signal processing algorithms in real-time.
- Reflect on and apply basic theories/design principles of persuasive technology (*e.g.*, see Consolvo *et al.*, CHI'06, Campbell *et al.*, CSCW'08).

Parts

- [6 pts] Design an **algorithm to robustly track steps** using the Android smartphone's built-in accelerometer (and, possibly gyroscope). At a minimum, your signal processing approach should include a *smoothing filter* and a *peak detection* algorithm. The algorithm should recognize a new step with <= 2 sec latency.
- [3 pts] Design **a debug visualization interfaces** that shows a line graph of the real-time accelerometer signal, the smoothed signal, and debug information about your tracker algorithm. This is part of your midpoint check-in due next week, see: https://canvas.uw.edu/courses/1199409/assignments/4187237. You can roll-your-own simple line graph or use an existing library (like GraphView).
- [2 pts] The **debug interface should also show**: (i) the number of steps your algorithm has tracked; (ii) the number of steps tracked by Android's *Step Detector Sensor*; (iii) and the number of steps tracked by Android's *Step Counter Sensor*. Update: it appears that the Huawei Honor7X phones do not support the Android Step Detector Sensor (see Discussion). So, you can compare with just the Step Counter Sensor. For those using your own devices, let us know if neither the Step Detector nor Step Counter sensor works on your device.
- [3 pts] Design a **creative feedback interface** that highlights both when a new step is sensed as well as provides a cumulative count. For example, perhaps you make a fractal tree that dynamically grows based on the number

Related Items
SpeedGrader™
Download Submissions
0 out of 1 Submissions Graded

QUESTIONS?

SCHEDULE TODAY: 6:30-9:20

06:35-07:00: Discuss the readings

07:00-08:00: Signal Processing in UbiComp

08:00-08:10: Short break

08:10-09:20: Work on A1

LEARNING GOALS

How do we **acquire sensor data** (*i.e.,* signal acquisition)?

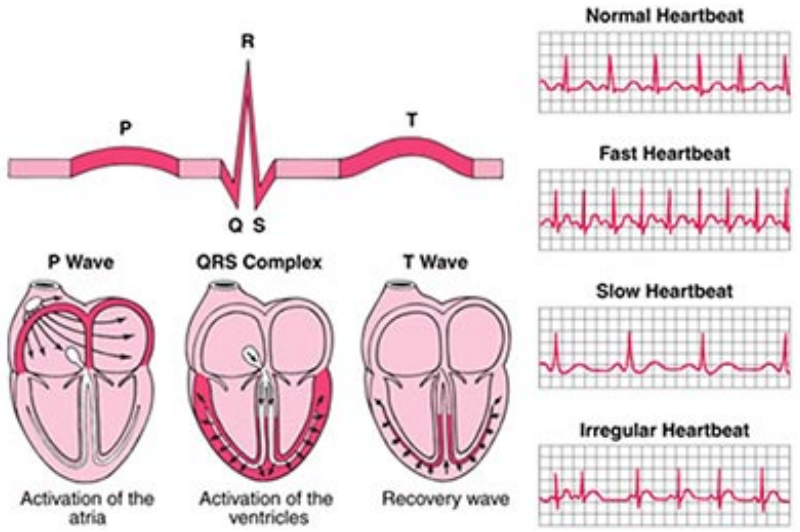
What is a time-series **signal** and how do we **represent them**?

Decomposing and synthesizing **signals**

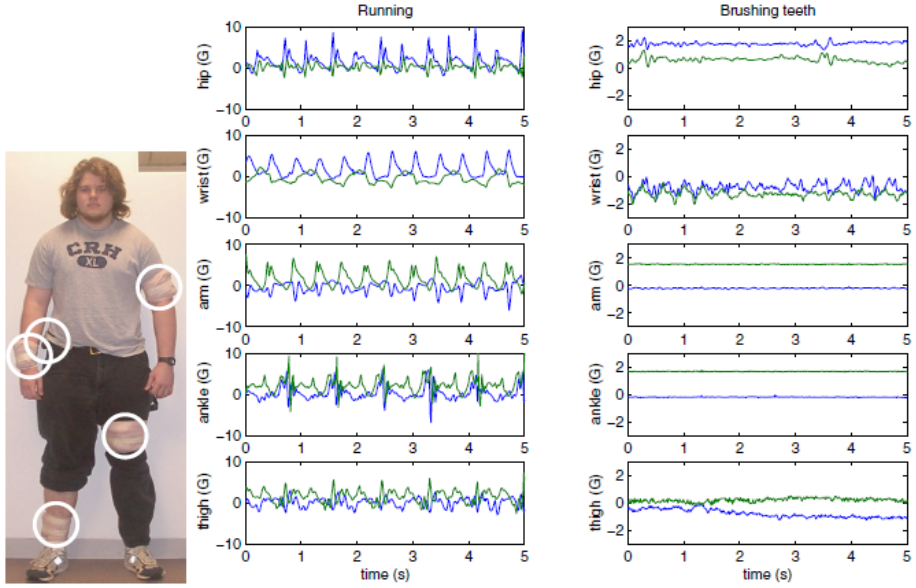
How to approach analyzing and processing **signals**

TIME-SERIES SIGNALS

Sensors *sense* physical phenomena and convert them to electrical signals, which are digitized and represented as time-series arrays

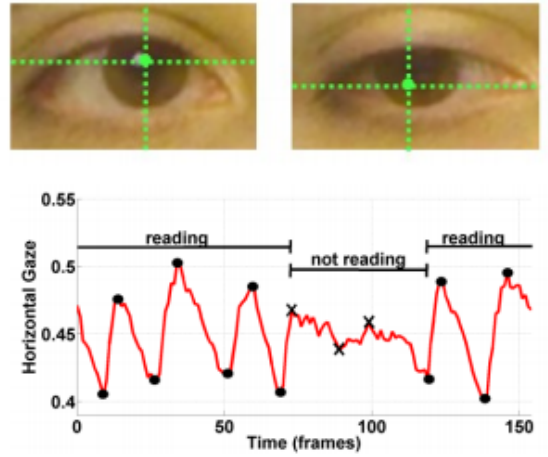


Electrocardiograms for Heartbeat Tracking



Source: Bao & Intille, Pervasive 2004

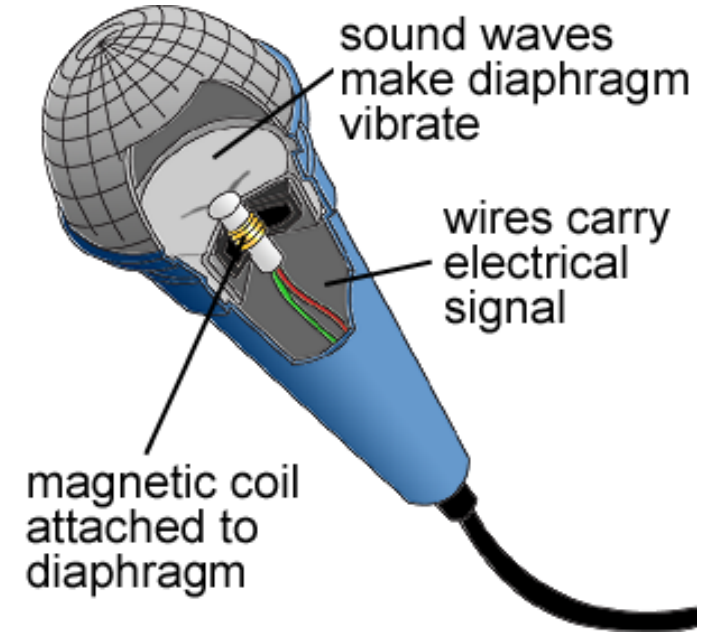
Accelerometers for Activity Recognition

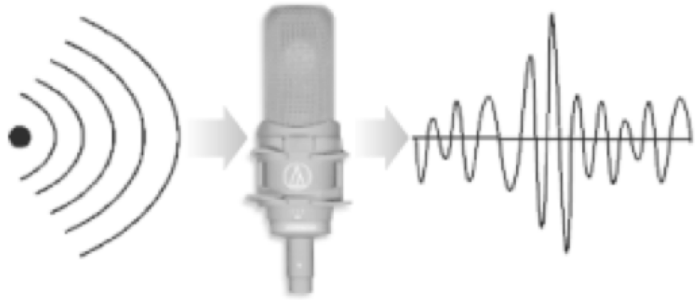


Source: Mariakakis *et a.*, CHI'15

Smartphone Camera for Gaze Tracking

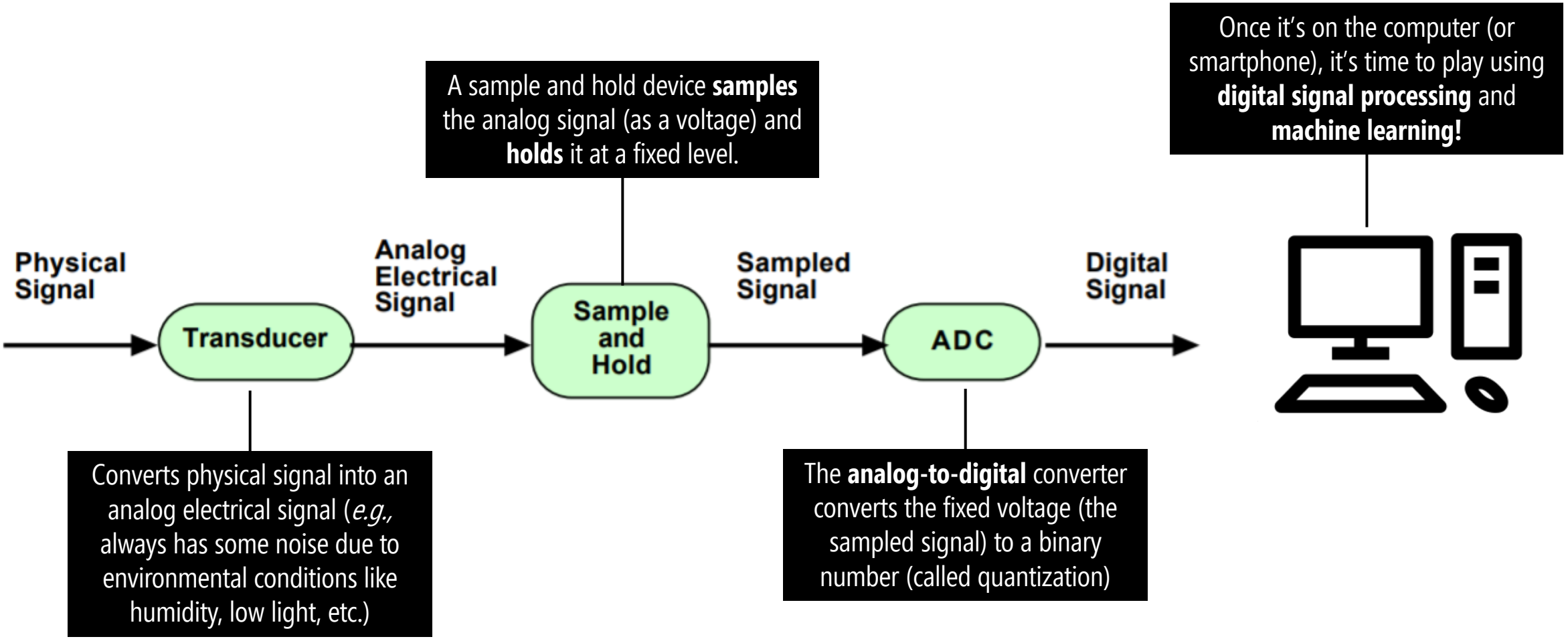
EXAMPLE: A MICROPHONE





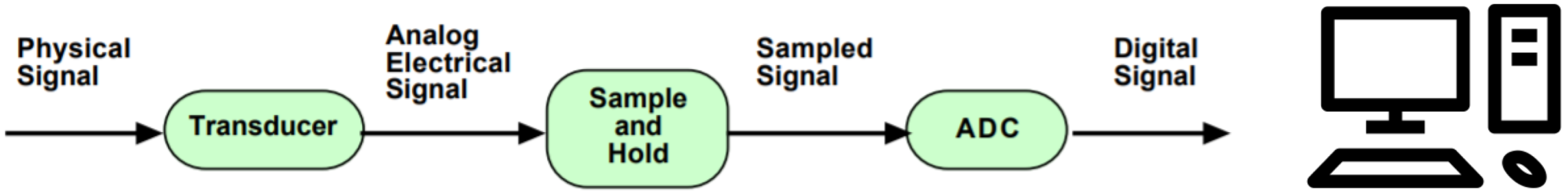
1. **Physical stimulus.** When you speak, play an instrument, *etc.*, it vibrates air in a sinusoidal pattern
2. **Diaphragm vibration.** Inside a microphone is a small diaphragm that vibrates in response to air movement
3. **Electricity generation.** A conductive coil, which is attached to the diaphragm moves back and forth across a magnet. This generates a current proportional to the sound wave.

SIGNAL ACQUISITION



Source: Stiber, Stiber, & Larson, Signal Computing, 2016 (link)

SIGNAL ACQUISITION: TWO KEY QUESTIONS



1. **Sampling rate:** How often do I need to sample the "real world?"
2. **Quantization**: How many bits should I use to represent each sample?

Source: Stiber, Stiber, & Larson, Signal Computing, 2016

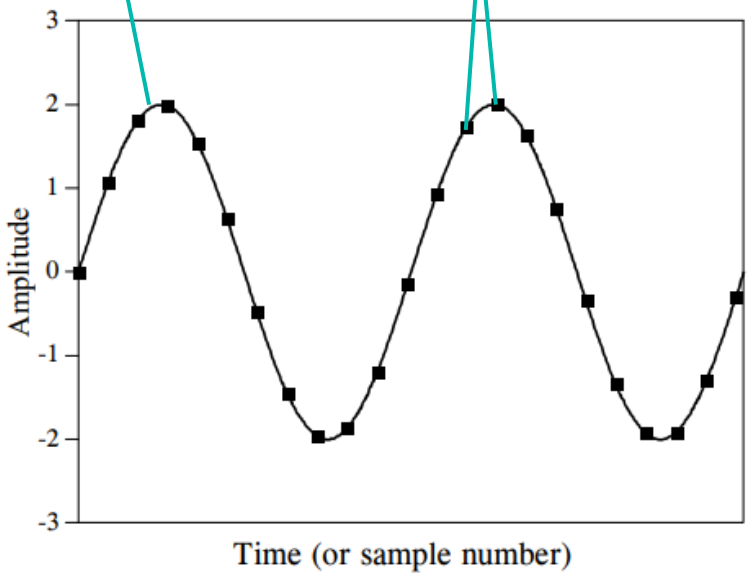
SIGNAL ACQUISITION

SAMPLING RATE

The rate that you sample the "real world." You should sample at a rate sufficient to accurately reconstruct the real analog signal.

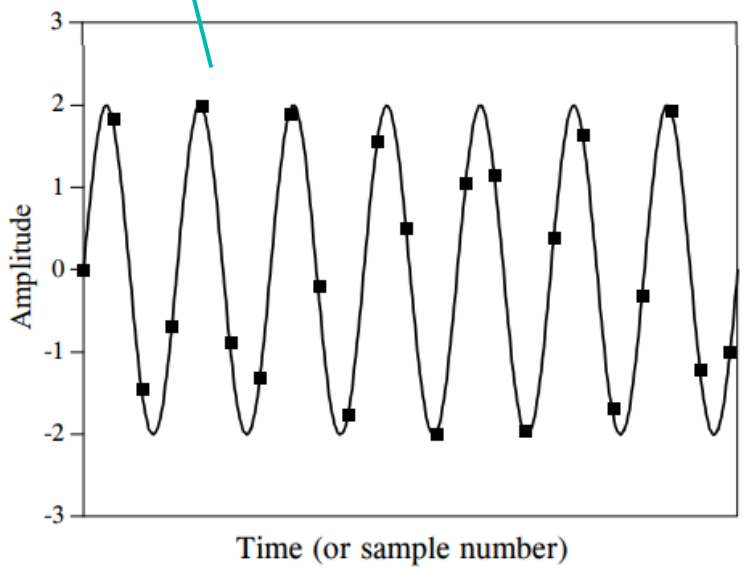
Solid line is the "real world" analog signal

Each dot represents a digital sample



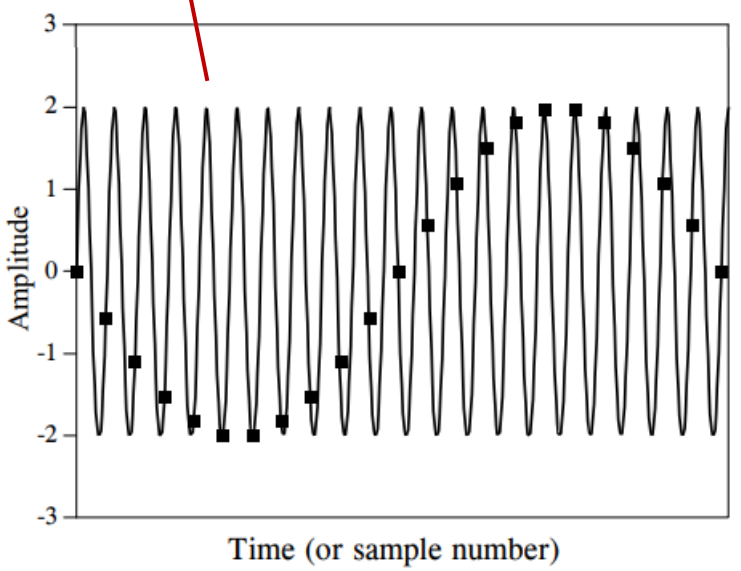
Real-world signal: 1 Hz
Sampling frequency: 12 Hz
Ratio: 1/12 ≈ 0.08

This one "looks bad" but there's actually enough information (mathematically) to properly reconstruct the real signal.



Real-world signal: 3.72 Hz
Sampling frequency: 12 Hz
Ratio: 3.72/12 ≈ 0.31

Uh, oh! The sampled data now looks like a different signal! This is called **aliasing**.



Real-world signal: 11.4 Hz
Sampling frequency: 12 Hz
Ratio: 11.4/12 = 0.95

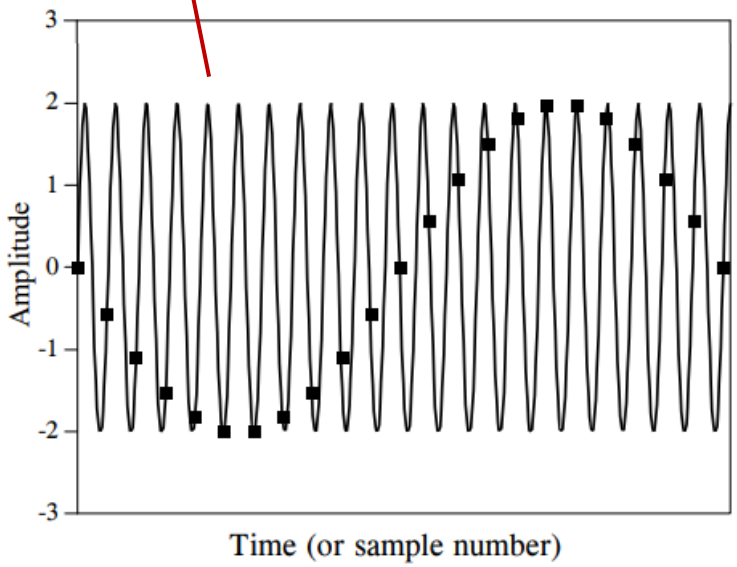
Source: Smith, The Scientist and Engineer's Guide to DSP, 1998 (link)

SIGNAL ACQUISITION

SAMPLING THEOREM

Frequently called the *Shannon* sampling theorem or the *Nyquist* sampling theorem

Uh, oh! The sampled data now looks like a different signal! This is called **aliasing**.



Real-world signal: 11.4 Hz
Sampling frequency: 12 Hz
Ratio: 11.4/12 = 0.95

The sampling theorem states that a continuous signal can only be properly sampled *if it does not contain frequency components above one-half of the sampling rate*.

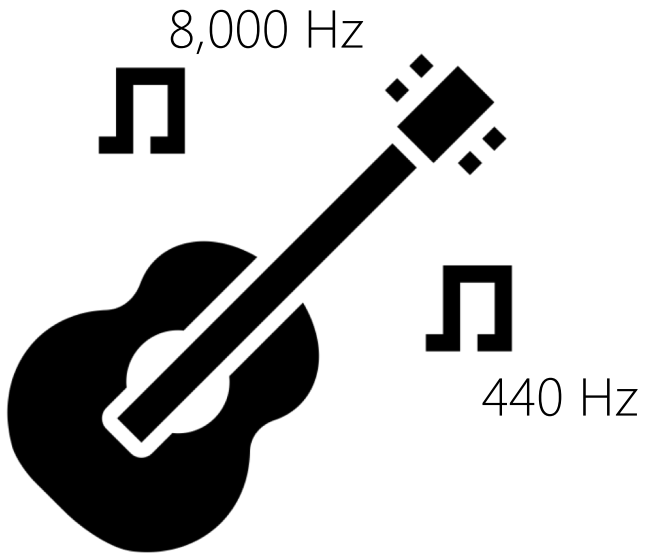
$$Sampling\ rate\ must\ be > 2 * \max(real\ world\ signal\ frequency)$$

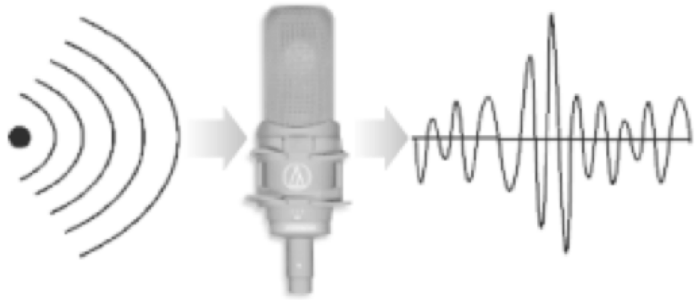
The **Nyquist frequency** is the minimum sampling rate you can in order to properly represent the signal

Source: Smith, The Scientist and Engineer's Guide to DSP, 1998 (link)

ALIASING EXAMPLE: RECORDING SOUND

When recording audio, if you play frequencies that are above ½ the sampling rate, you will get aliasing!





Sensor (Microphone)
Sampling frequency: 4,000 Hz

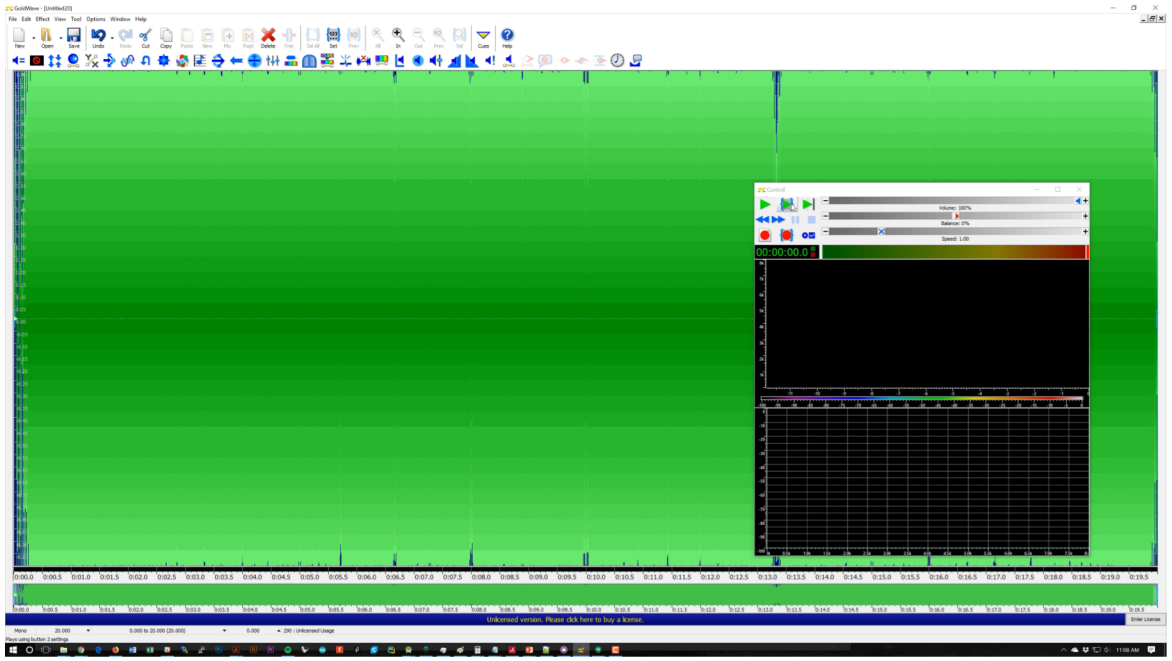
A **sampling rate of 4,000Hz** requires that any analog signal (in this case, sound) be composed of **frequencies 2,000** and below.

Any **frequencies** present in the signal **above this limit** will be **aliased** to frequencies between 0-2,000 Hz and will be combined with whatever information was legitimately there.

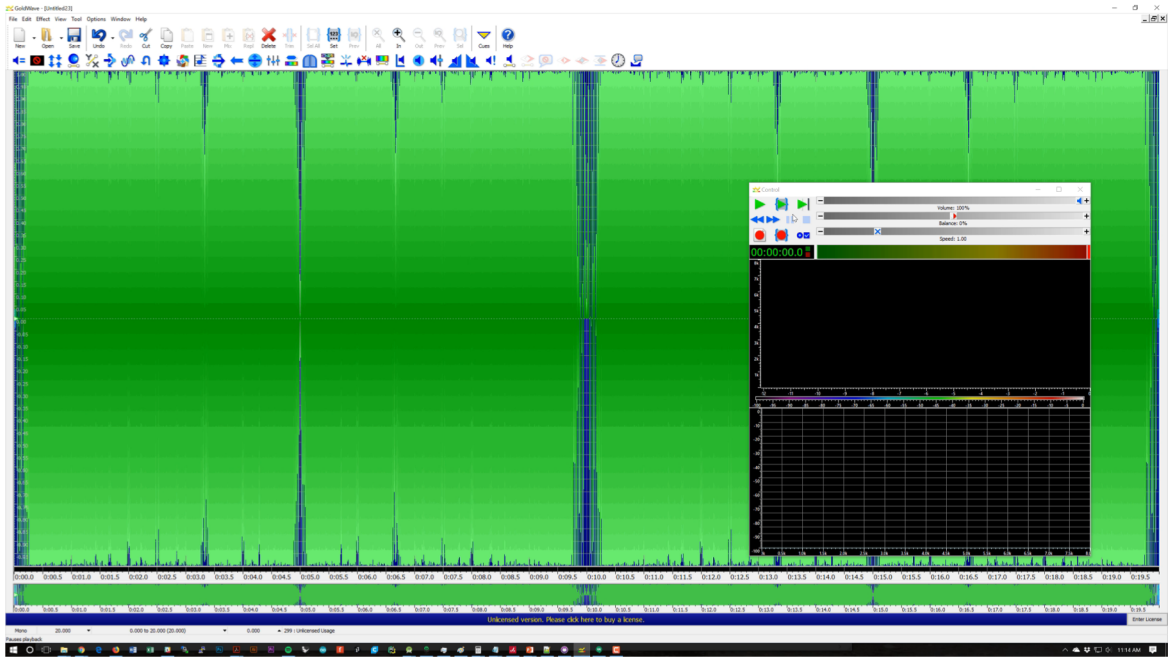
See: Smith, The Scientist and Engineer's Guide to DSP, 1998 (link)

ALIASING EXAMPLE: RECORDING SOUND

When recording audio, your sampling rate must be 2x the largest frequency in your “real-world” signal



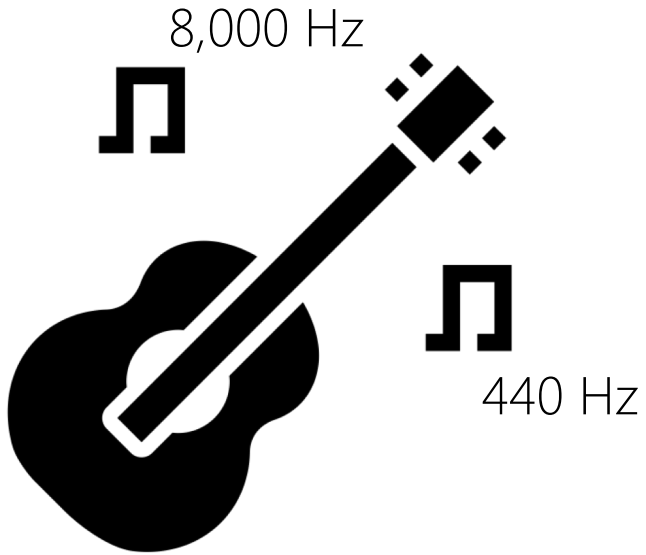
“Real-world” signal: 0Hz – 8,000 Hz (frequency sweep)
Sampling frequency: 16,000 Hz
Ratio: 8,000/16,000 = 0.5



“Real-world” signal: 0Hz – 8,000 Hz (frequency sweep)
Sampling frequency: 4,000 Hz
Ratio: 8,000/4,000 = 2 (ratio needs to be 0.5 or below)

See: Smith, The Scientist and Engineer’s Guide to DSP, 1998 (link)

ALIASING EXAMPLE: RECORDING SOUND



The range of human hearing is ~20Hz – 20kHz with far more sensitivity between 1-4kHz

What sampling rate should we record music at then?

The **sampling theorem** provides the answer. The sampling rate needs to be at least 2 x 20kHz or 40kHz. CDs, for example, use 44.1kHz. Note: a majority of speech occurs < 8,000Hz so can use lower sampling rates when recording speech only.

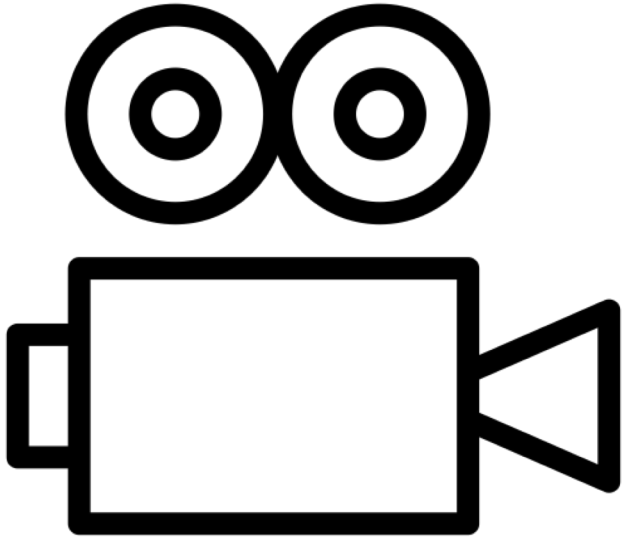
See: Smith, The Scientist and Engineer's Guide to DSP, 1998 (link); http://www1.cs.columbia.edu/~hgs/audio/44.1.html

SIGNAL ACQUISITION

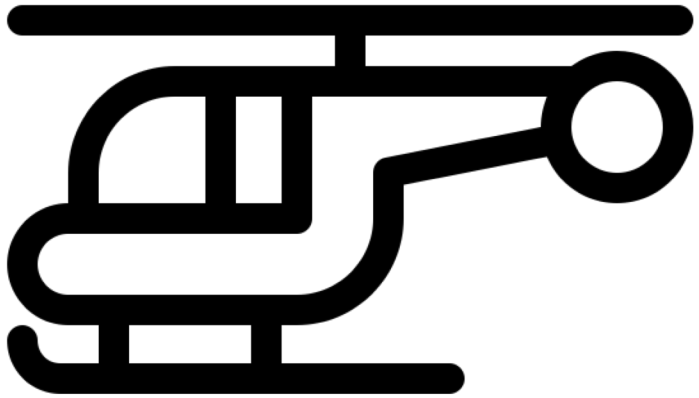
ALIASING EXAMPLE: VIDEO RECORDING MOVEMENT

Source: https://youtu.be/yr3ngmRuGUc; See also: https://youtu.be/AYQAKwCxScc

ALIASING EXAMPLE: CAMERAS



Video cameras typically record at:
24Hz – 60Hz (*i.e.,* 24 – 60 fps)



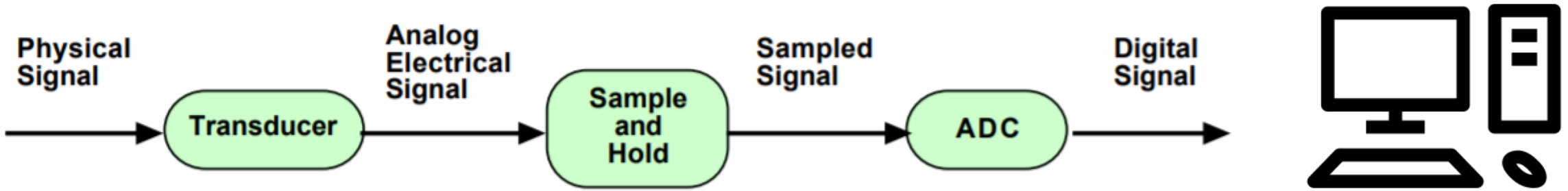
How fast do helicopter blades spin?
250 – 600 rpms

How fast would we need to record video to capture 600 rpm blades?

Minimum sampling rate
= 2 x 600 = 1,200 fps
(which would allow 2 captures per blade spin; more would be necessary for fluid video)

See: Smith, The Scientist and Engineer's Guide to DSP, 1998 (link); http://www1.cs.columbia.edu/~hgs/audio/44.1.html

SIGNAL ACQUISITION: TWO KEY QUESTIONS



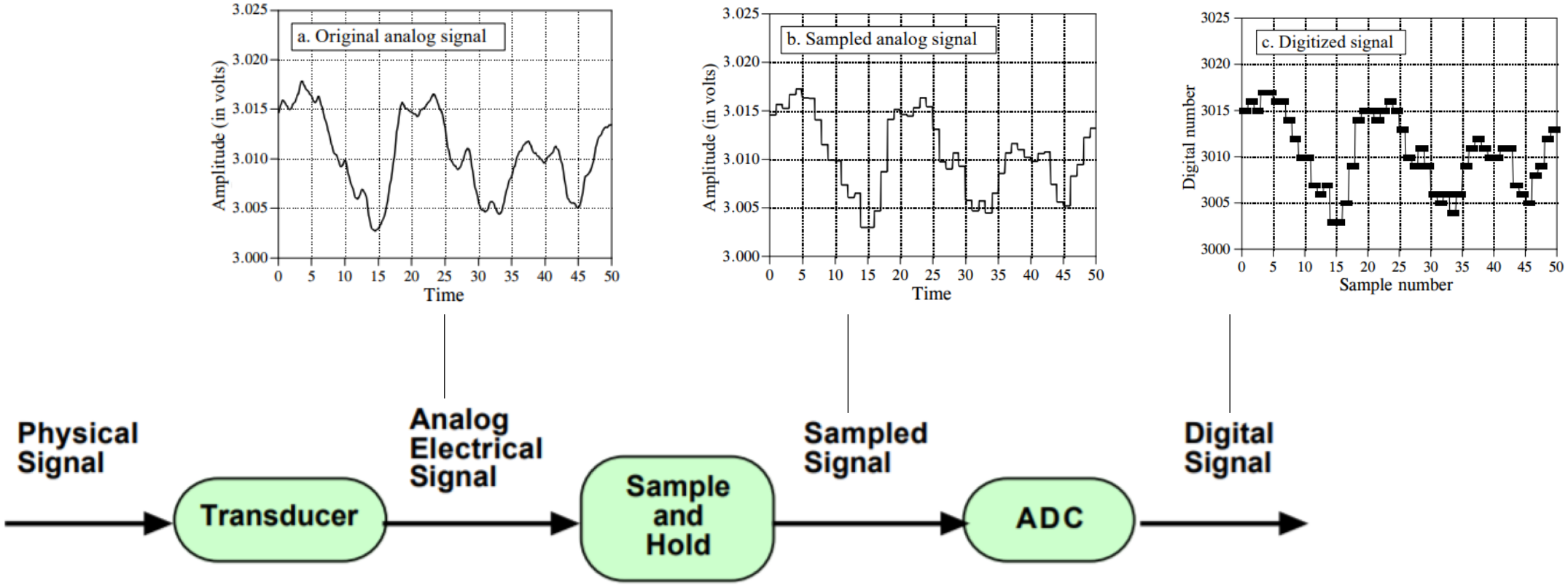
1. **Sampling rate:** How often am I sampling the "real world"?
2. **Quantization**: How many bits are used to represent each sample?

Source: Stiber, Stiber, & Larson, Signal Computing, 2016 (link)

SIGNAL ACQUISITION

QUANTIZATION

Quantization is the process of digitizing an analog signal. Quantization inherently involves rounding (and thus creates noise).

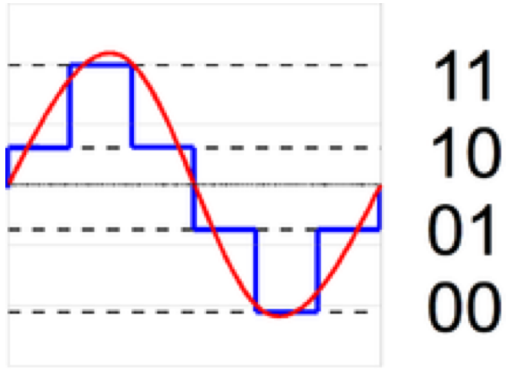


See: Smith, The Scientist and Engineer's Guide to DSP, 1998 (link); Stiber, Stiber, & Larson, Signal Computing, 2016 (link). Images from Wikipedia.

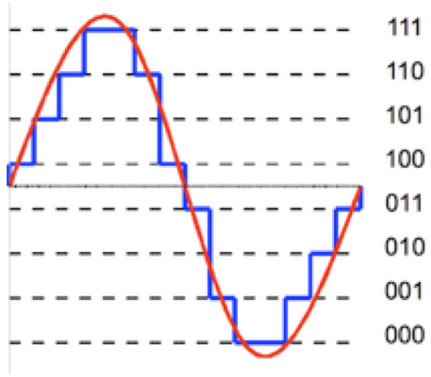
SIGNAL ACQUISITION

QUANTIZATION

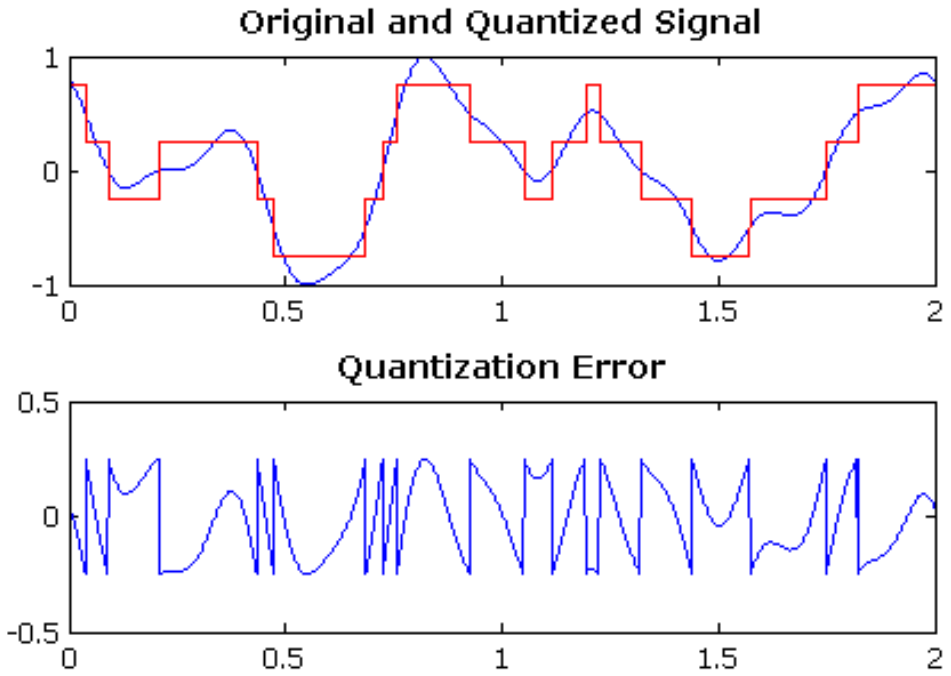
Quantization is the process of digitizing an analog signal. Quantization inherently involves rounding (and thus creates noise).



2-Bit Quantization
2-bit resolution allows for four discretization levels of the analog signal



3-Bit Quantization
3-bit resolution allows for eight discretization levels of the analog signal

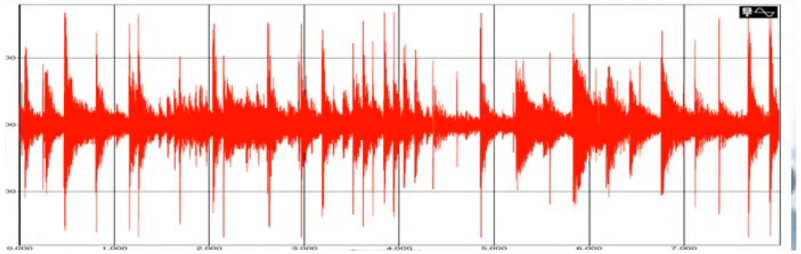


Quantization Error
The quantization error is the difference between an input value and its quantized value

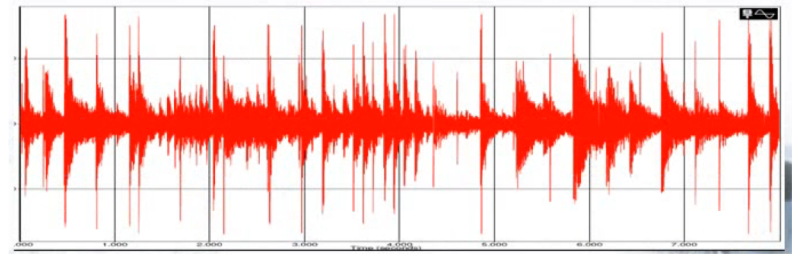
See: Stiber, Stiber, & Larson, Signal Computing, 2016 (link). Images from Wikipedia.

QUANTIZATION EXAMPLE: AUDIO

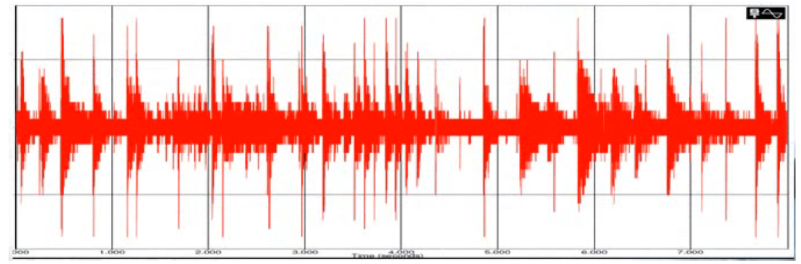
24 BITS
16+ million levels



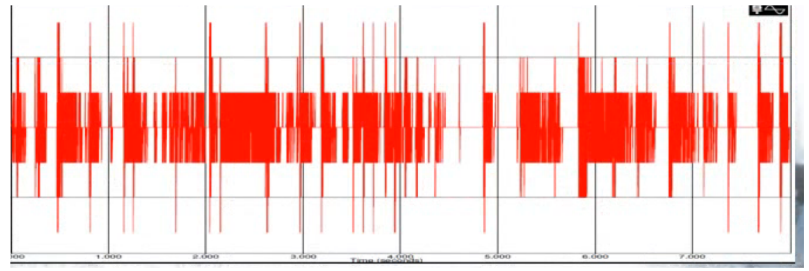
8 BITS
256 levels



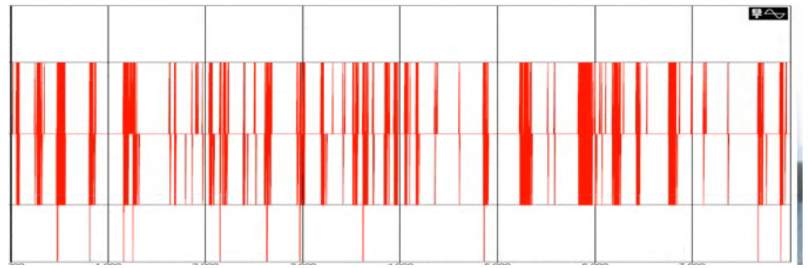
5 BITS
32 levels



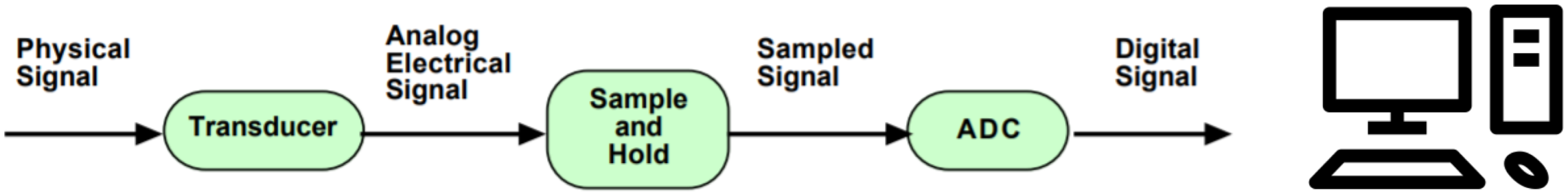
3 BITS
8 levels



2 BITS
4 levels



Source: https://youtu.be/_cRFBBnUFug



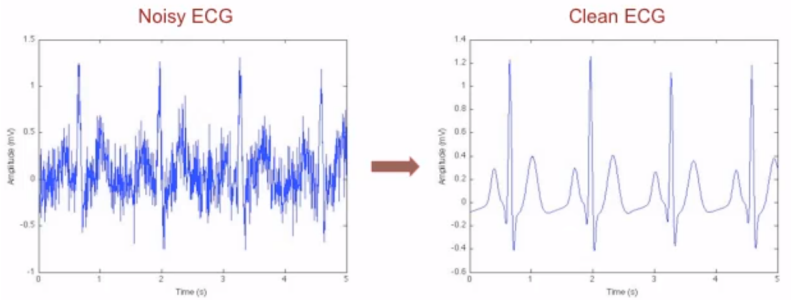
Processing the digital signal
(aka digital signal processing or DSP)

WHAT IS SIGNAL PROCESSING?

Manipulating/processing a signal to change its characteristics or to extract information (*e.g.,* filtering, transforming, correlating).

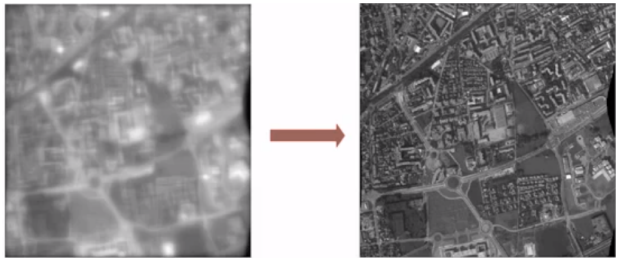
ELIMINATING NOISE

(*e.g.,* smoothing ECG)



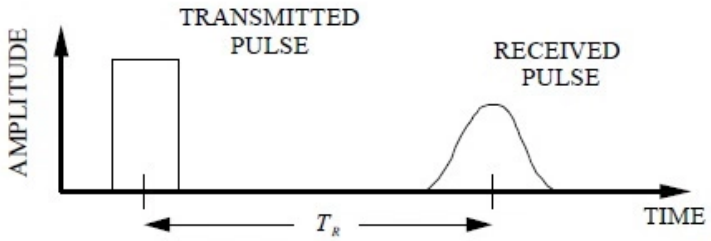
CORRECTING DISTORTION

(*e.g.,* Hubble lens)



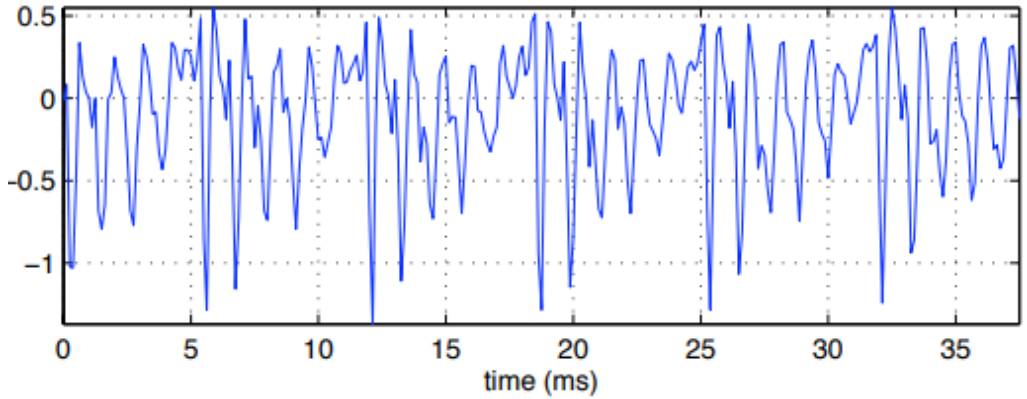
EXTRACTING INFORMATION

(*e.g.,* distance & velocity from radar pulse)

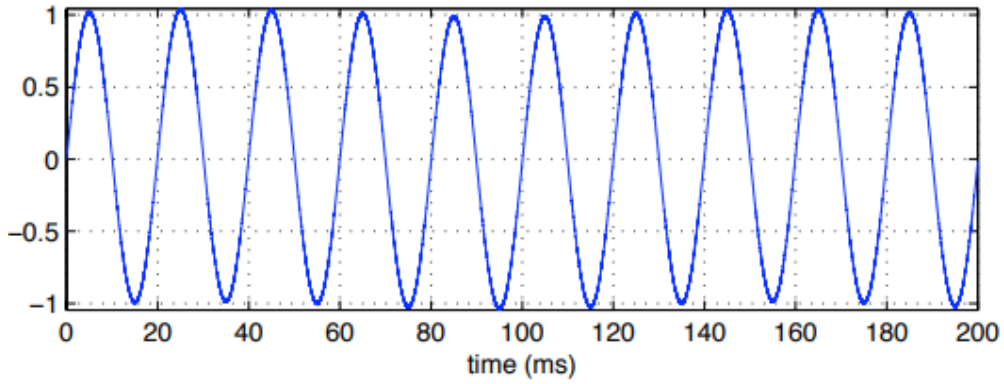


See: Barry Van Veen's website: http://allsignalprocessing.com; Images from: https://youtu.be/YmSvQe2FDKs

BREAKING DOWN A SIGNAL: WHAT DO YOU SEE?



The human speech of the vowel ‘a’ as in ‘bat.’

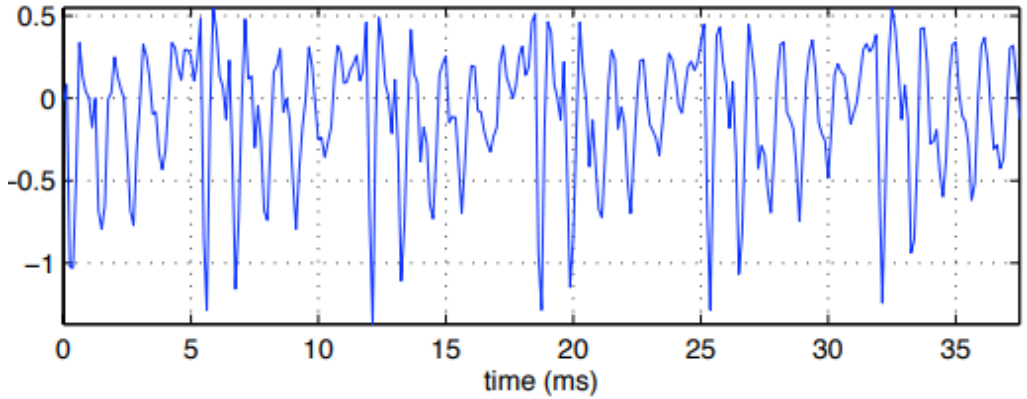


The ringing of a tuning fork

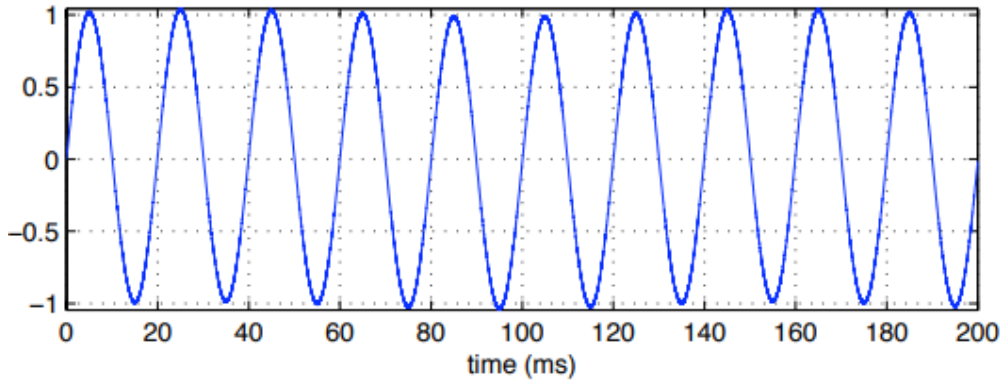
Source: Stiber, Stiber, & Larson, Signal Computing, 2016 (link)

BREAKING DOWN A SIGNAL: WHAT DO YOU SEE?

These time-series signals are called *periodic* signals because they repeat. It turns out that any periodic signal can be represented as a sum of related sines and cosines, which forms the basis of all signal processing.



The human speech of the vowel 'a' as in 'bat.'



The ringing of a tuning fork

$$Periodic\ signals\ are\ represented\ by\ g(t) = \sin(\omega t + \theta) = \sin(2\pi f t + \theta)$$

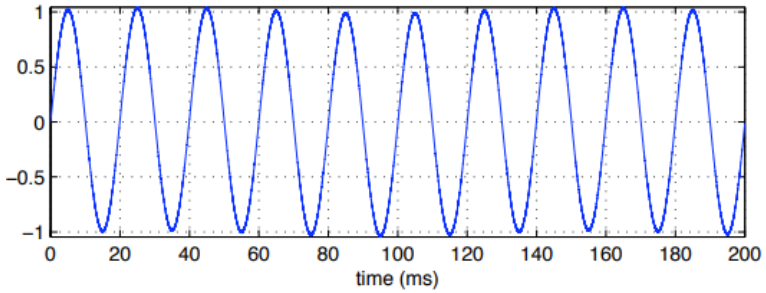
$$Cosine\ is\ just\ sin\ phase - shifted\ by\ \frac{\pi}{2}\colon \cos(\omega t) = \sin(\omega t + \frac{\pi}{2})$$

Source: Stiber, Stiber, & Larson, Signal Computing, 2016 (link)

REPRESENTATIONS OF SINUSOIDS

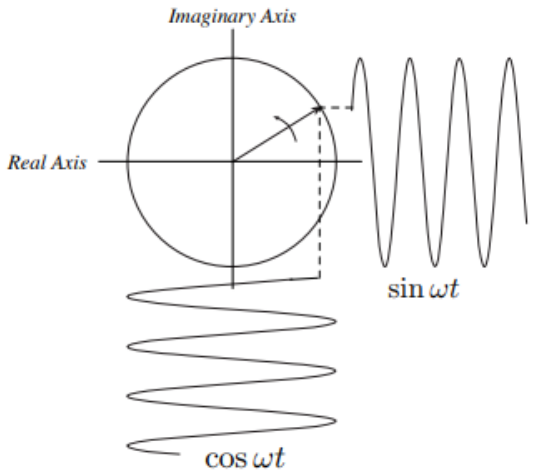
Rather than thinking of a sinusoid as a function which oscillates up and down, you can think of it as something that goes round and round. This is the phasor representation, which links to the complex exponential representation.

SINUSOIDAL REPRESENTATION



$$g(t) = \sin(\omega t + \theta)$$

PHASOR REPRESENTATION



COMPLEX EXPONENTIAL

$$e^{j\theta(t)} = e^{j\omega t}$$

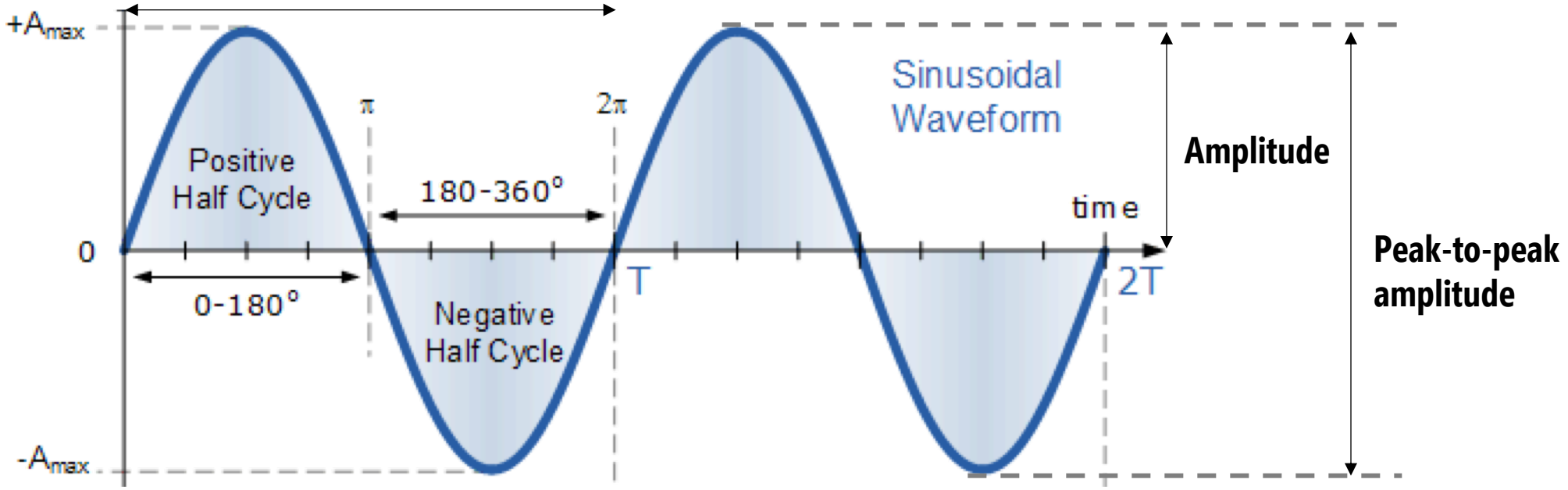
Any periodic function can be written as a sum of phasors represented by these complex exponentials. A super powerful idea used in lots signal processing (like Fourier transforms!).

Source: Stiber, Stiber, & Larson, Signal Computing, 2016 (link)

SINUSOIDS

One full cycle or period of waveform.

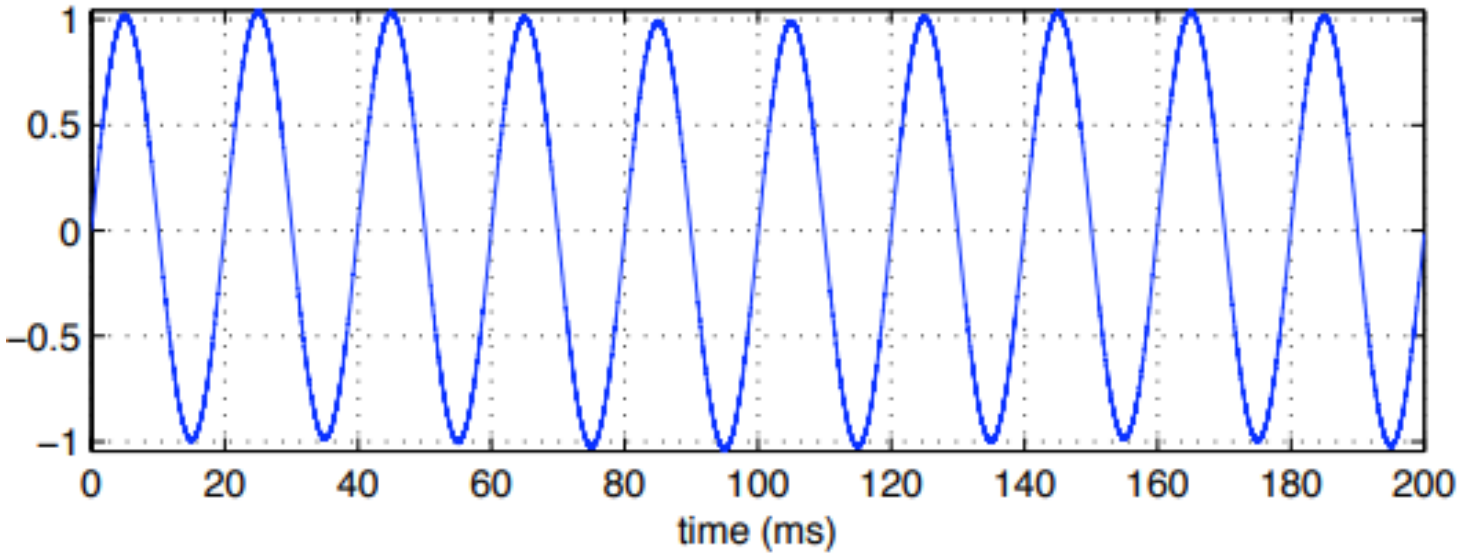
$$Period\ T = \frac{1}{f}$$



$$g(t) = \sin(\omega t + \theta) = \sin(2\pi f t + \theta)$$

Source: Stiber, Stiber, & Larson, Signal Computing, 2016 (link)

SINUSOIDS



$$g(t) = \sin(\omega t + \theta) = \sin(2\pi f t + \theta)$$

Do this **individually** first. I will then ask you to pair up with someone next to you to share and discuss your answer.

Think, Pair, Share

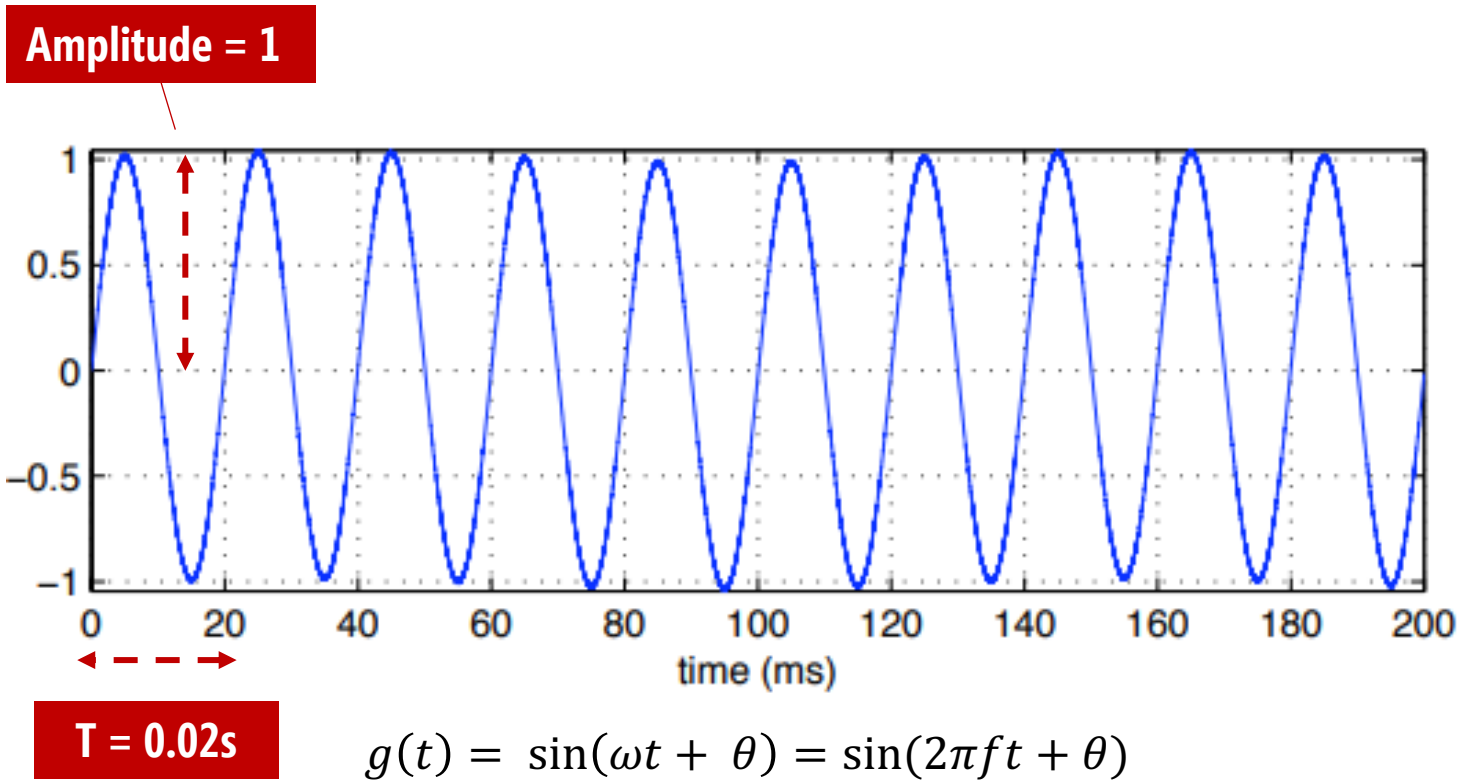
What is the amplitude of this signal?

What is the period (T) of the signal?

What is the frequency (*f*) of this signal?

Image source: Stiber, Stiber, & Larson, Signal Computing, 2016 (link)

SINUSOIDS



$$g(t) = \sin(\omega t + \theta) = \sin(2\pi f t + \theta)$$

Think, Pair, Share

What is the amplitude of this signal?

What is the period (T) of the signal?

What is the frequency (*f*) of this signal?

One full cycle or period of waveform.

$$Period\ T = \frac{1}{f}$$

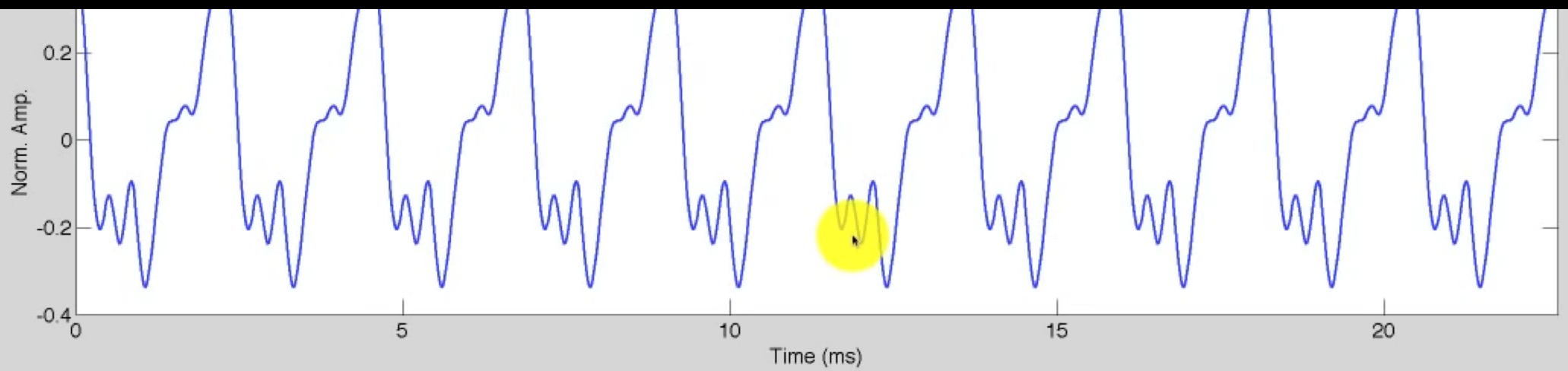
Thus

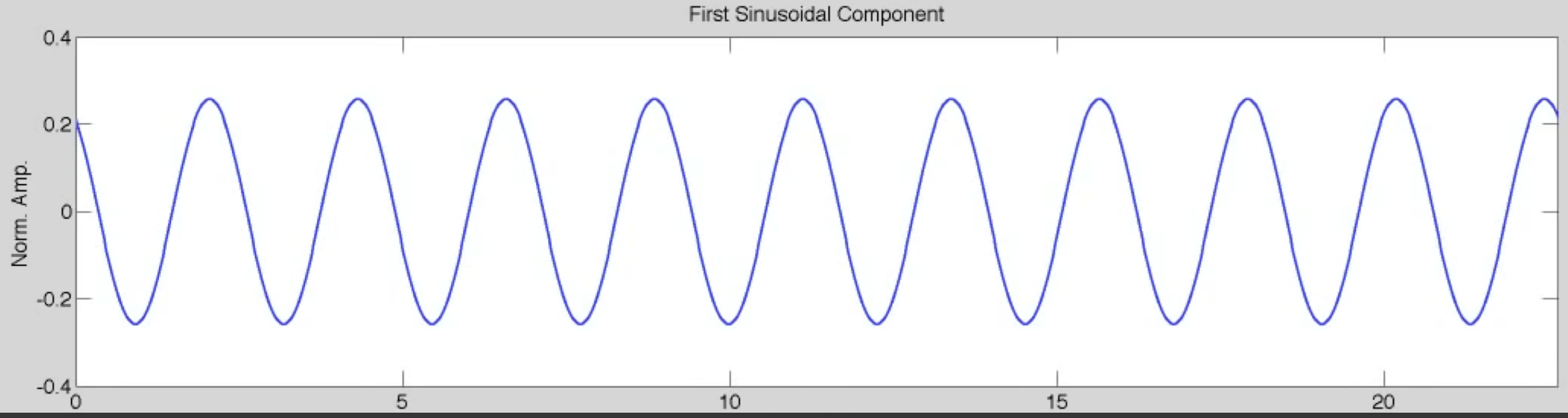
$$Frequency\ f = \frac{1}{T} = \frac{1}{0.02} = 50\ Hz$$

Image source: Stiber, Stiber, & Larson, Signal Computing, 2016 (link)

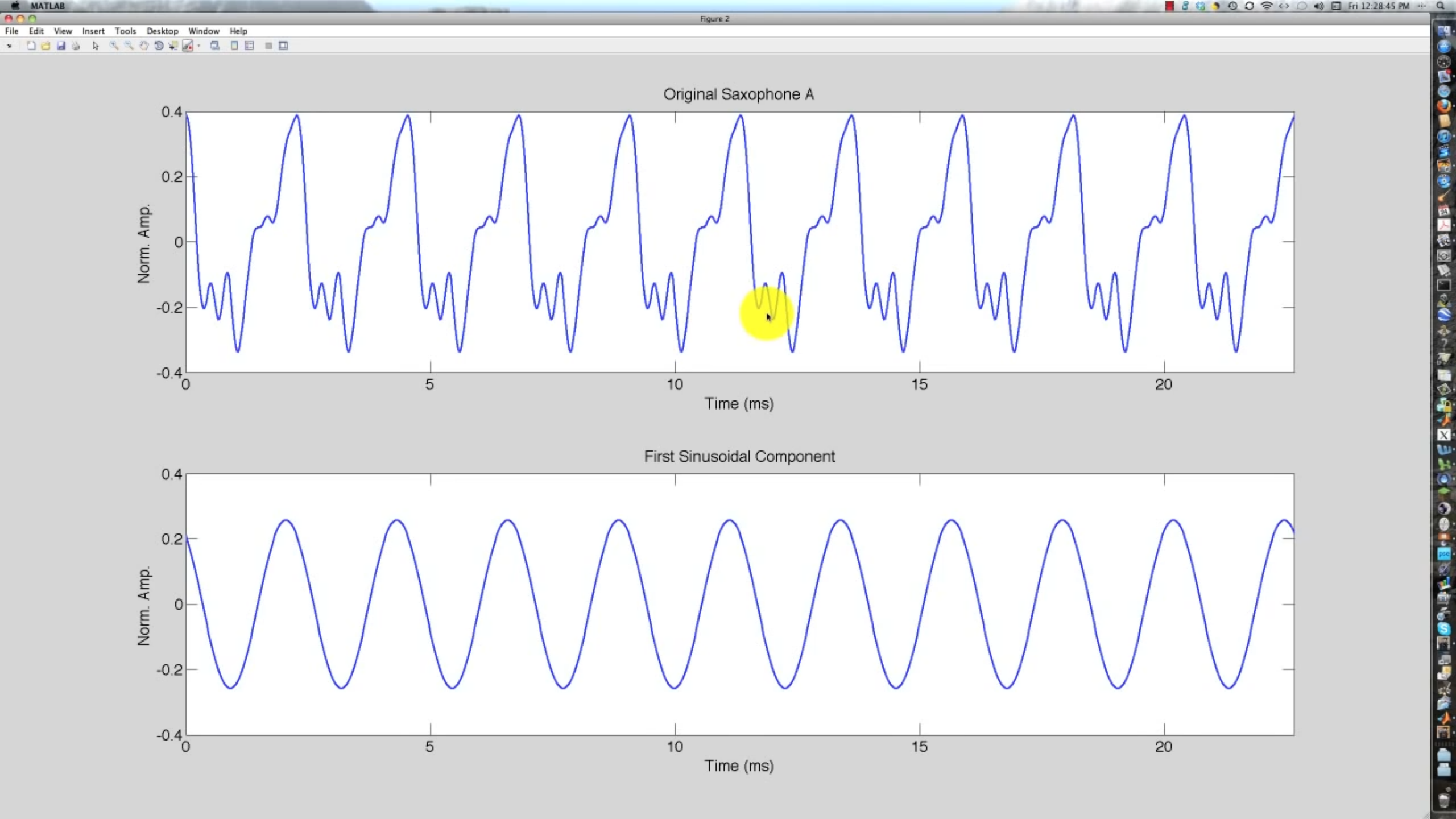
INTRODUCTION TO SIGNALS & DSP

REPRESENTING COMPLEX PERIODIC SIGNALS BY SUMMING SINUSOIDS

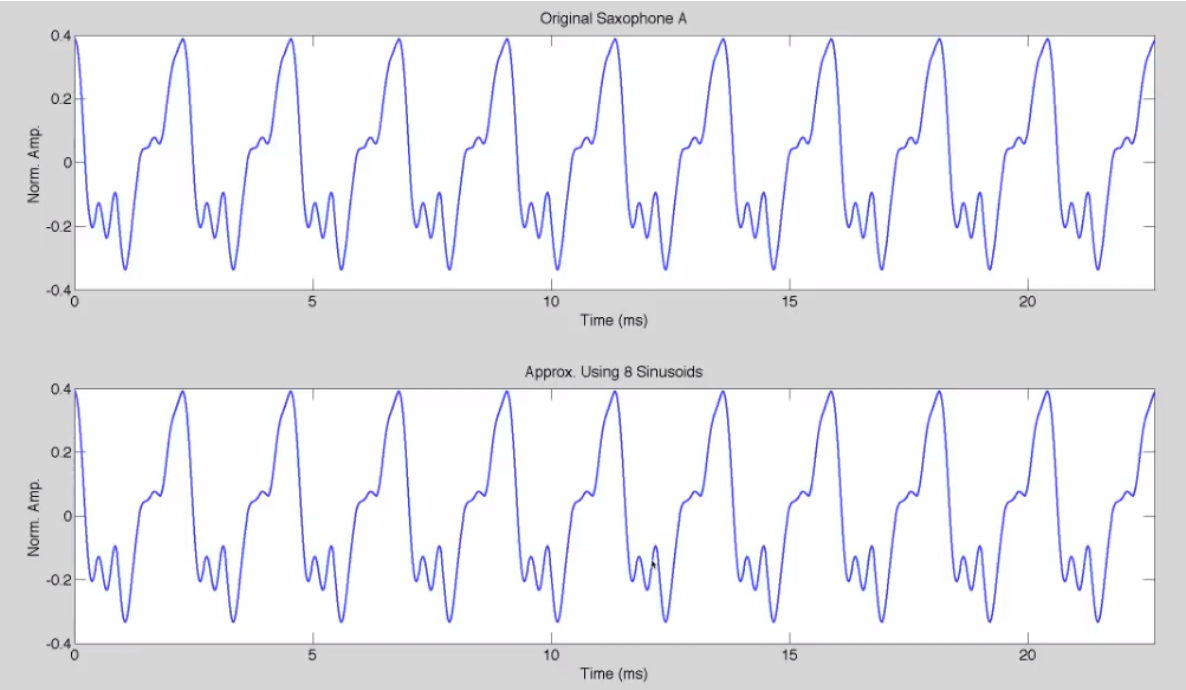




Source: Barry Van Veen's http://AllSignalProcessing.com. Direct video link: https://youtu.be/fCAZ7jcO-vc?t=10m43s


MATLAB
Figure 2
File Edit View Insert Tools Desktop Window Help
Original Saxophone A
Norm. Amp.
0.4
0.2
0
-0.2
-0.4
0
5
10
15
20
Time (ms)
First Sinusoidal Component
Norm. Amp.
0.4
0.2
0
-0.2
-0.4
0
5
10
15
20
Time (ms)

HOW'D HE DO THAT?



$$
\begin{aligned}
&A_1 * \sin(2\pi f_1 t + \theta) \\
&+ A_2 * \sin(2\pi f_2 t + \theta) \\
&+ A_3 * \sin(2\pi f_3 t + \theta) \\
&+ A_4 * \sin(2\pi f_4 t + \theta) \\
&+ A_5 * \sin(2\pi f_5 t + \theta) \\
&+ A_6 * \sin(2\pi f_6 t + \theta) \\
&+ A_7 * \sin(2\pi f_7 t + \theta) \\
&+ A_8 * \sin(2\pi f_8 t + \theta)
\end{aligned}
$$

Where f_n is an integer multiple of f_1 and A_n gets decreasingly small

SYNTHESIS AND DECOMPOSITION

This is an example of synthesis—combining signals by *scaling* (multiplications of signals by constants like A) and *addition*

Synthesis
Two or more signals are added together to form another signal.

Decomposition
Opposite from synthesis. Break one signal into two or more additive component signals.

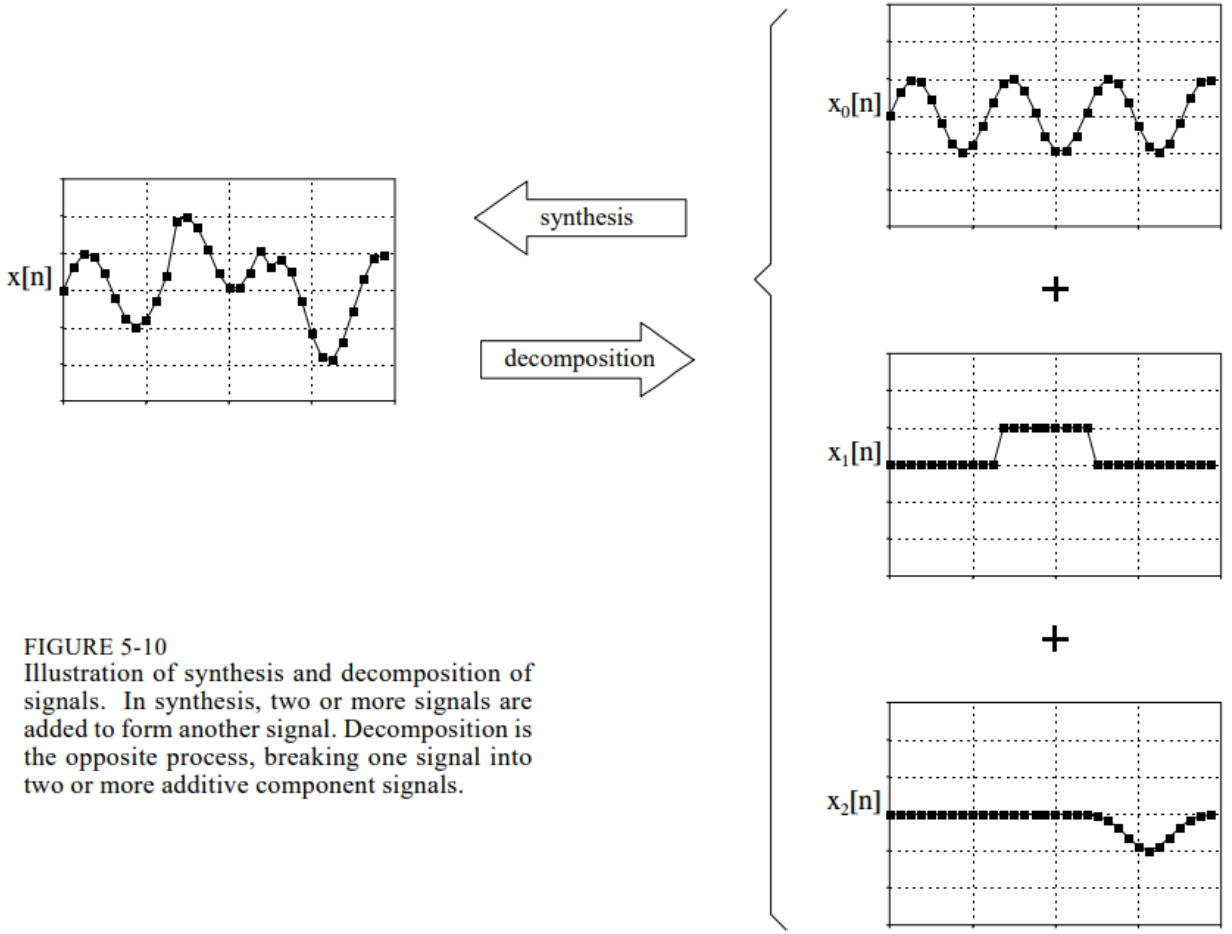
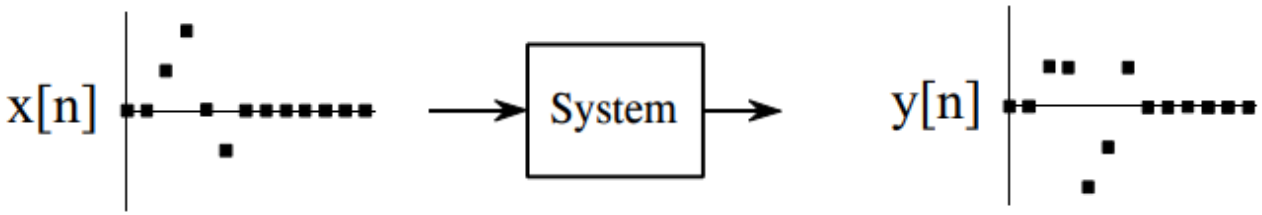


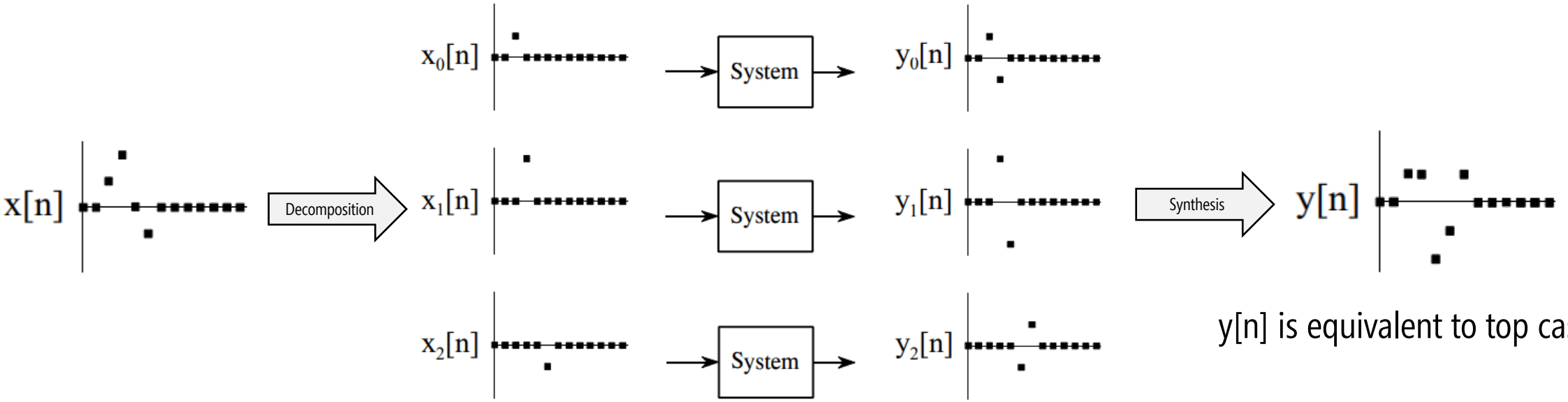
FIGURE 5-10
Illustration of synthesis and decomposition of signals. In synthesis, two or more signals are added to form another signal. Decomposition is the opposite process, breaking one signal into two or more additive component signals.

See: Chapter 5 of Smith, The Scientist and Engineer's Guide to DSP, 1998 (link); note this only works for linear systems

SUPERPOSITION!

Perhaps the most fundamental concept: passing all individual components of x[n] through a linear system (*e.g.,* a filter) produces output signals that, when synthesized (via addition), these output signals form the same signal produced when x[n] is passed through the system. This is the basis of nearly all signal processing techniques.





y[n] is equivalent to top case

See: Chapter 5 of Smith, The Scientist and Engineer's Guide to DSP, 1998 (link); Superposition only holds for linear systems


TIRED.
JUST TIRED.

LEARNING GOALS

How do we **acquire sensor data** (*i.e.,* signal acquisition)?

What is a time-series **signal** and how do we **represent them**?

Decomposing and synthesizing **signals**

How to approach analyzing and processing **signals**

SIGNAL PROCESSING: A UBICOMP APPROACH

Signal processing originated in the military with mathematical and electrical engineering roots (*e.g.,* sonar, radar). UbiComp is more applied.

1. **Capture 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 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.

DSP IN UBICOMP: TWO EXAMPLES

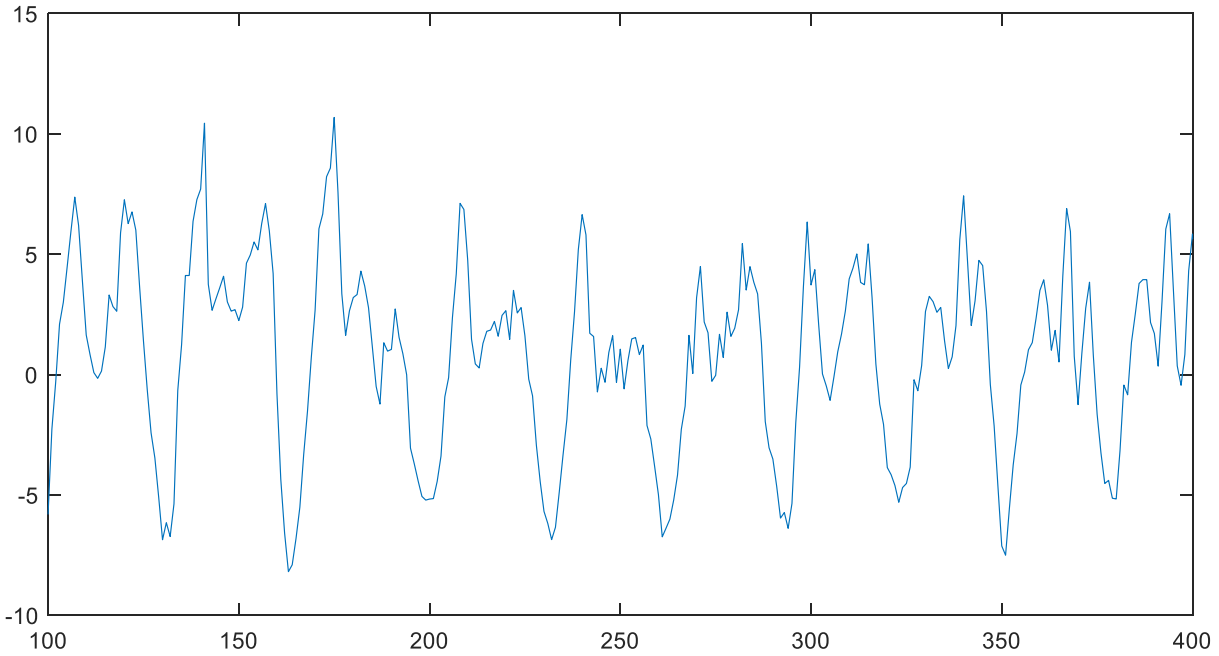
Sensing water usage using pressure waves

Step tracking using accelerometers ☺

How can we sense and identify **water usage activity** in the home?

HydroSense

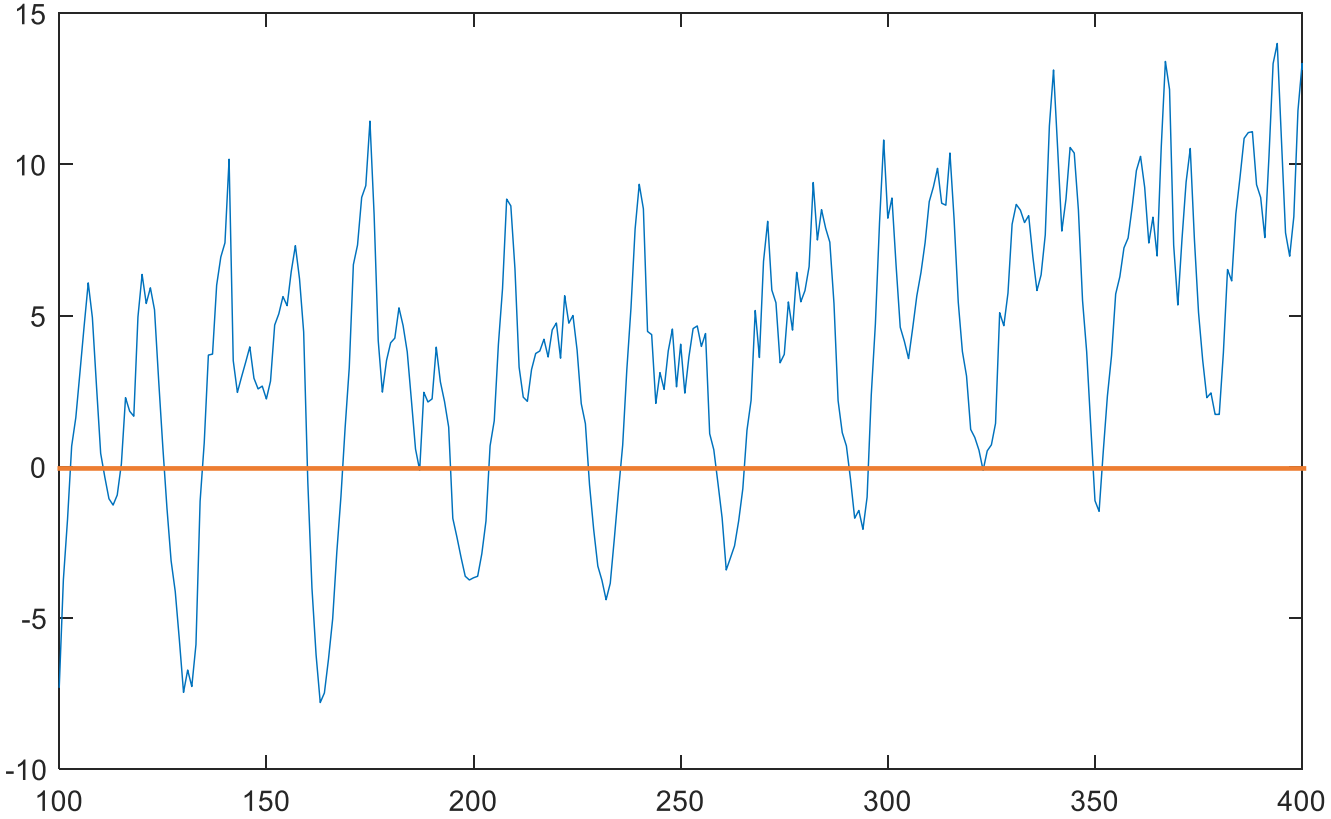
STEP TRACKER ACCELERATION SIGNAL



Think, Pair, Share

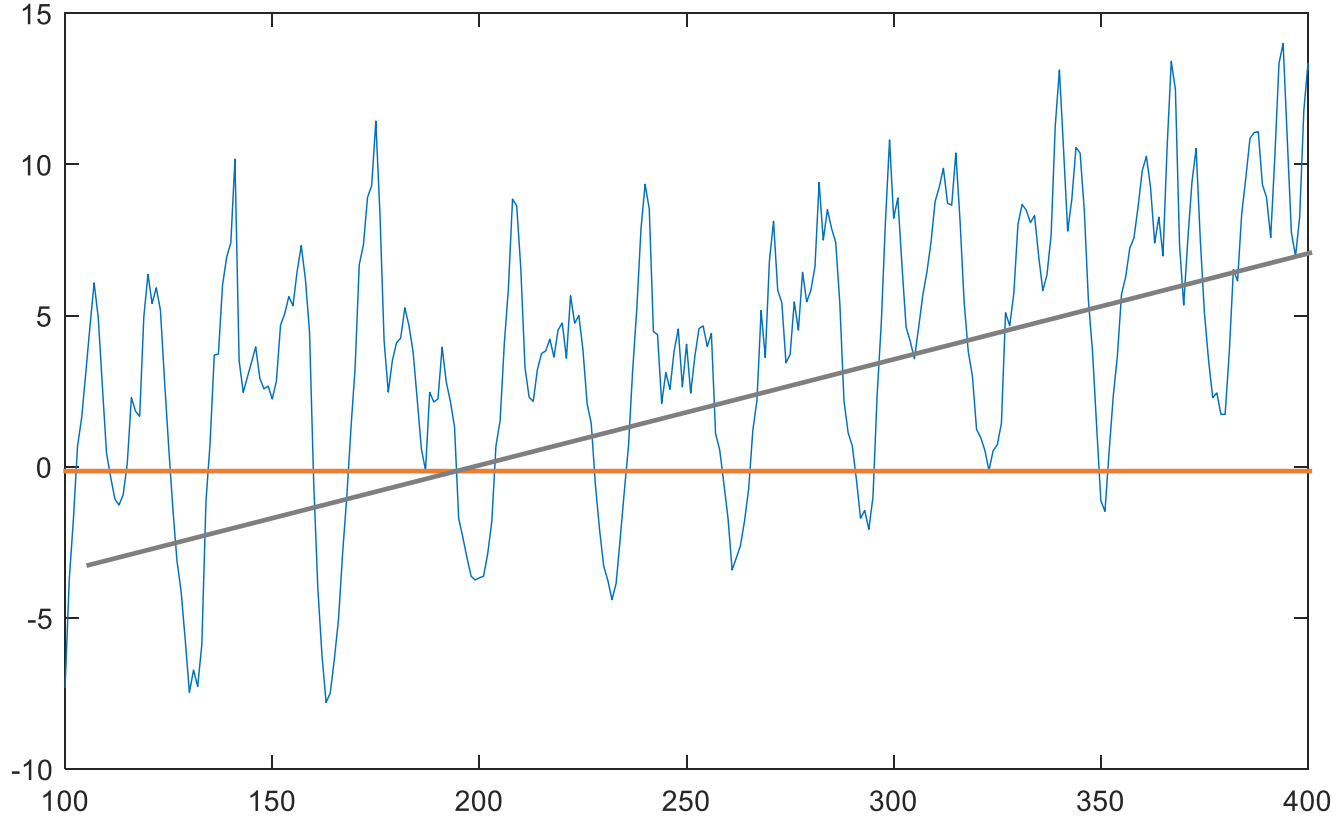
Take out a piece of paper and analyze this signal. Brainstorm and write down patterns that you observe. Then share them with a partner and discuss.

ONE WAY TO COUNT STEPS: DETECT ZERO CROSSINGS?



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

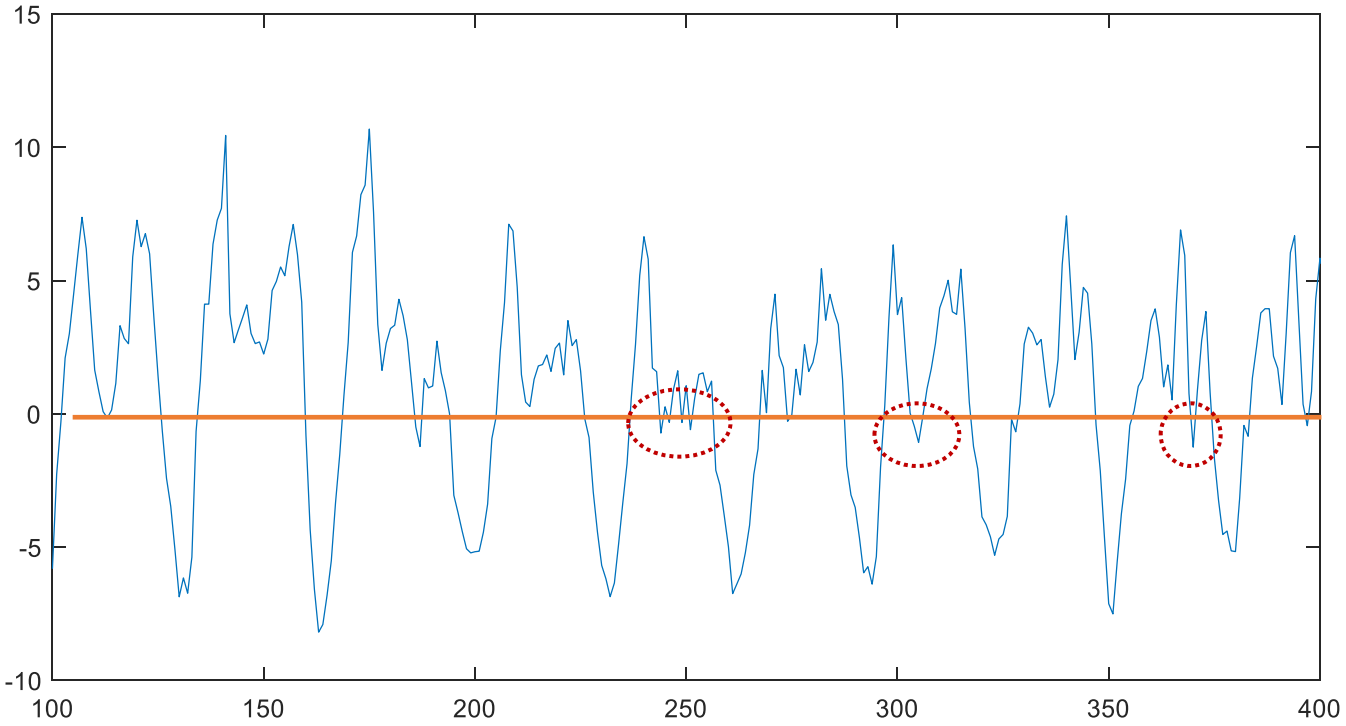
ONE WAY TO COUNT STEPS: DETECT ZERO CROSSINGS?



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.

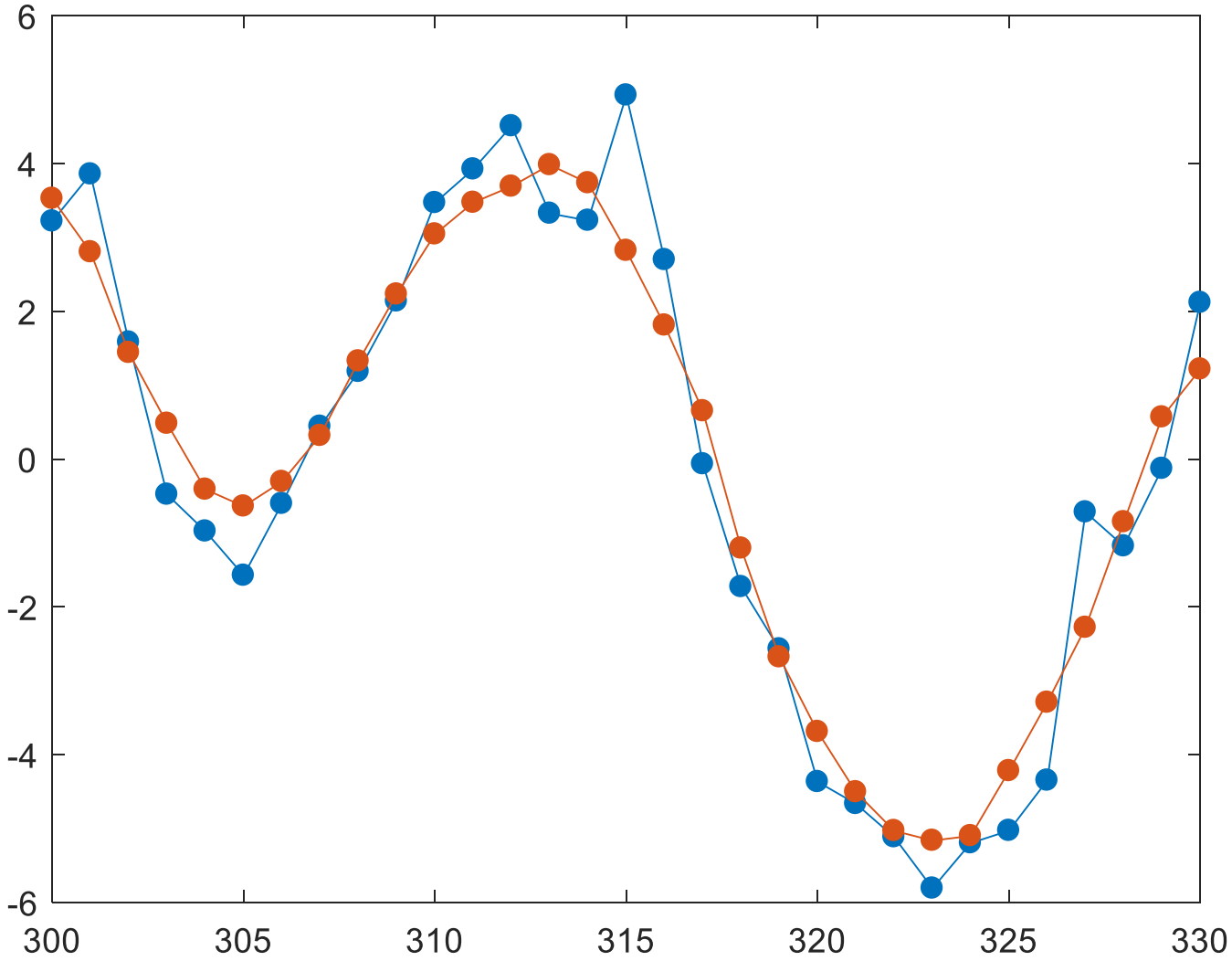
ONE WAY TO COUNT STEPS: DETECT ZERO CROSSINGS?



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)

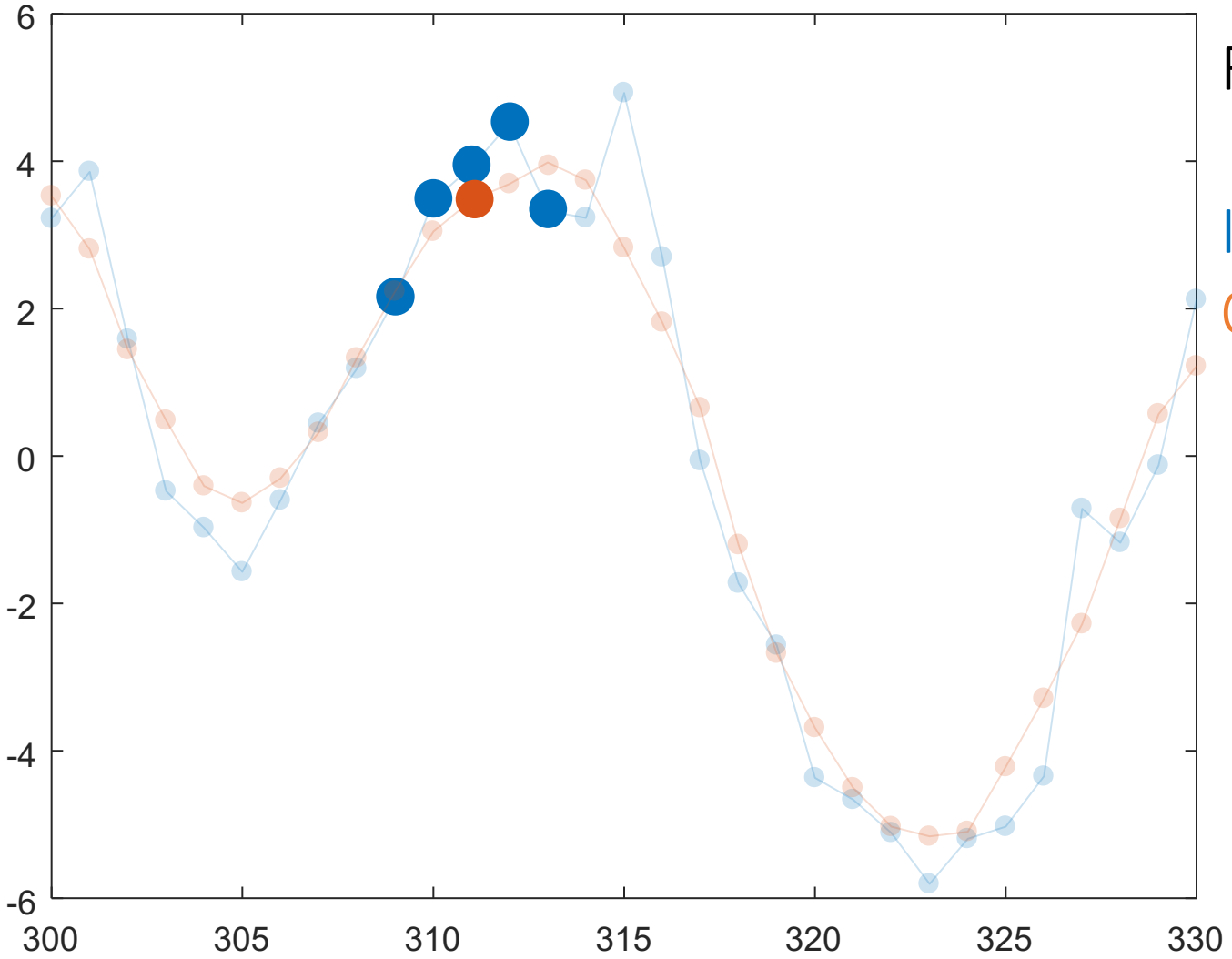
MOVING AVERAGE FILTER



Noisy Signal

Filtered Signal

MOVING AVERAGE FILTER

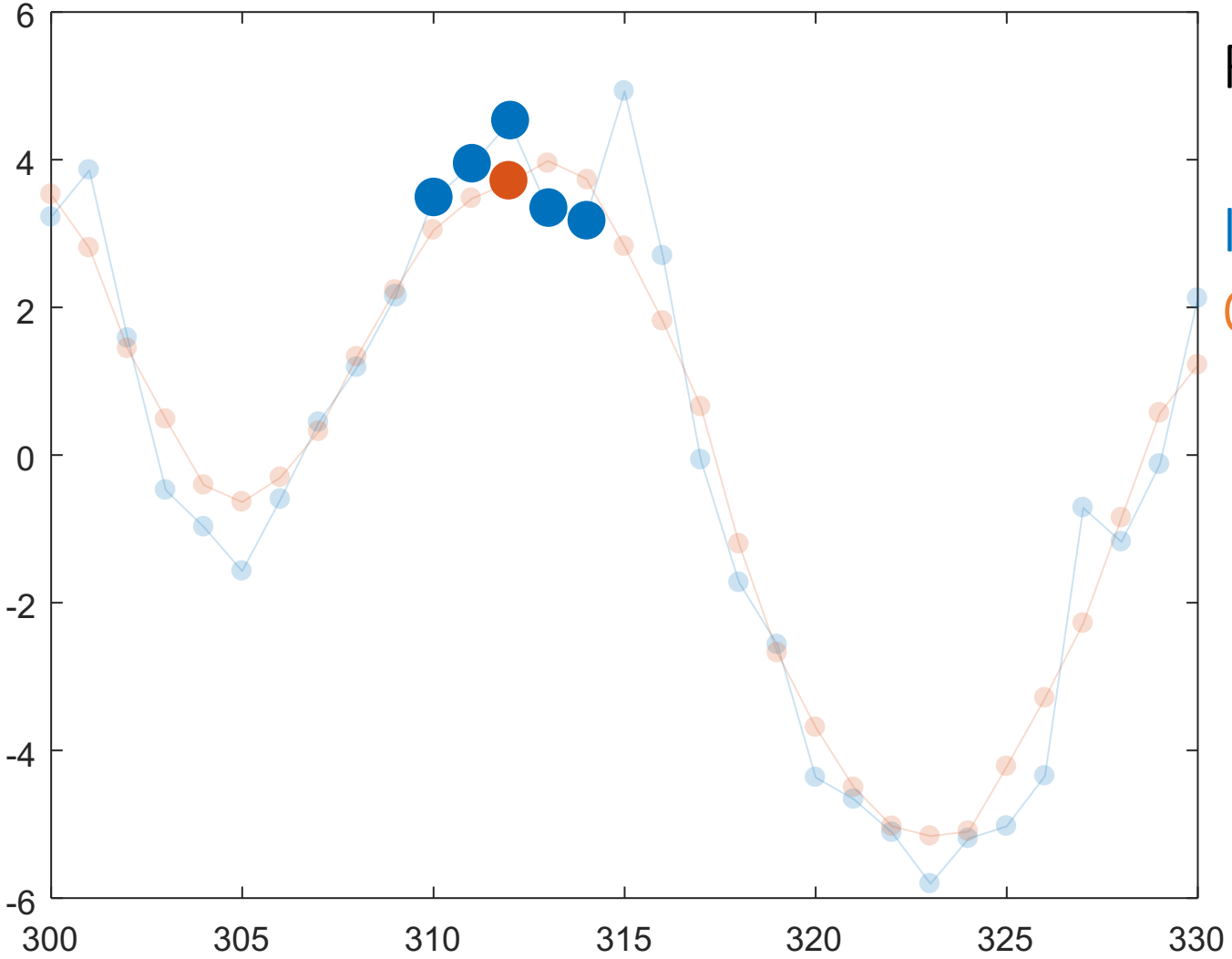


Filter Length [N] = 5

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

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

MOVING AVERAGE FILTER

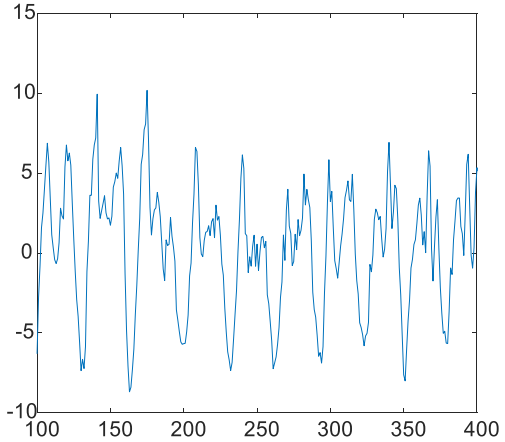


Filter Length [N] = 5

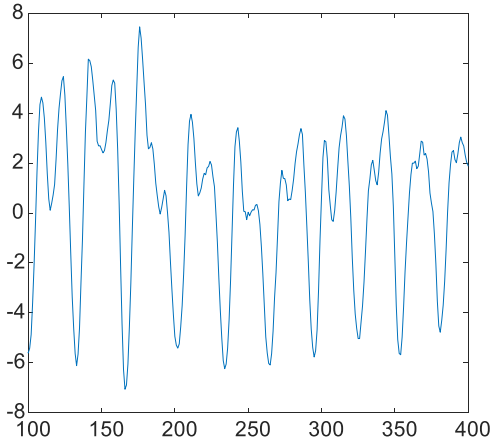
Input = [4,4.2,5,3.5,3.2]

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

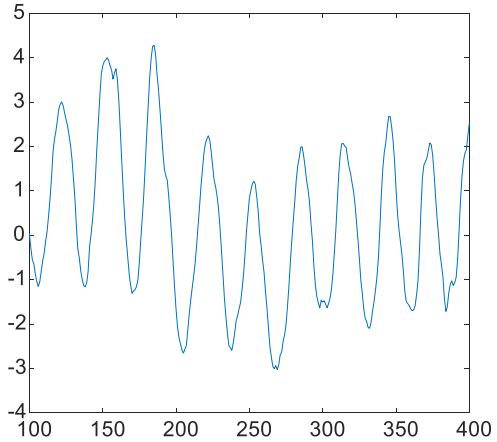
MOVING AVERAGE FILTER: DIFFERENT WINDOW SIZES



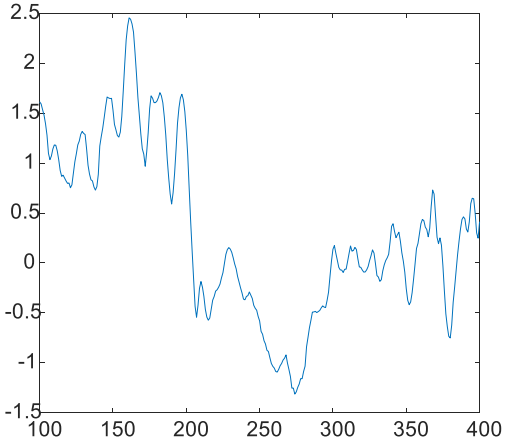
N = 1



N = 5

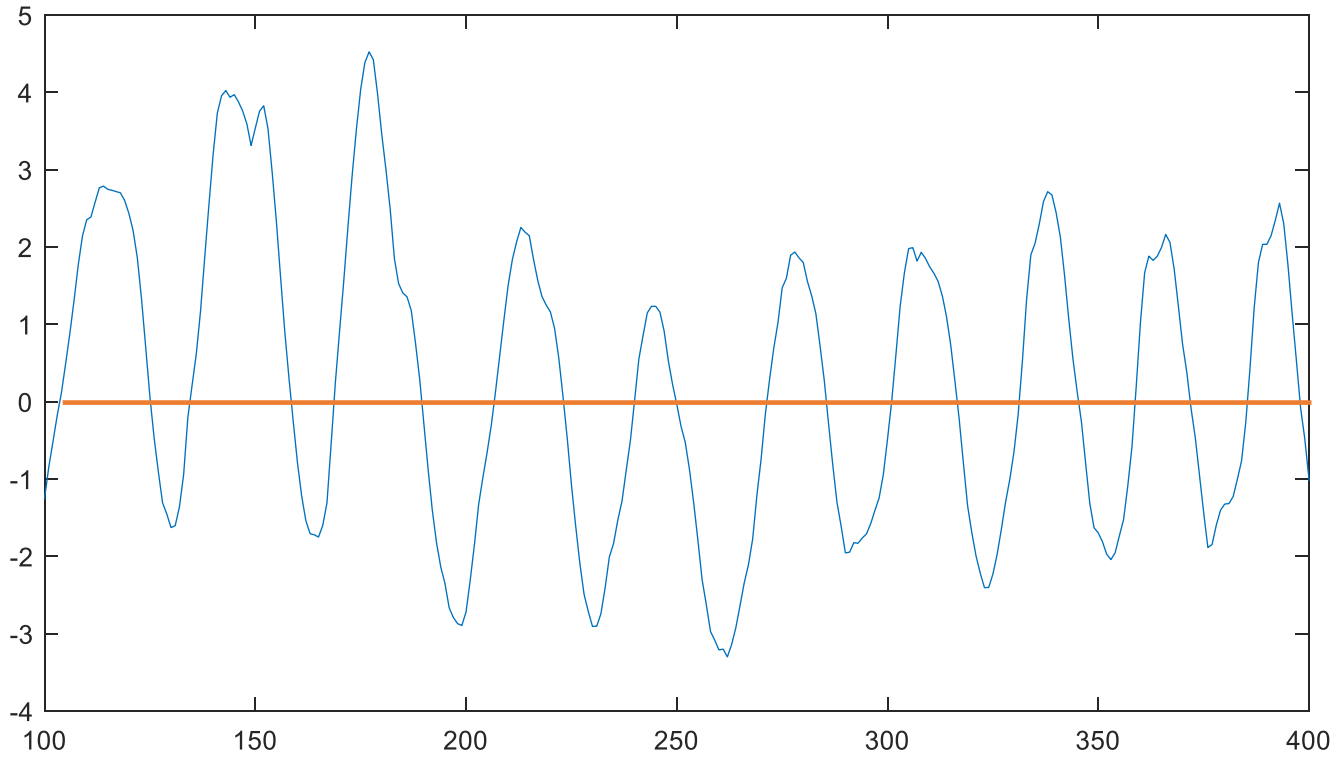


N = 15



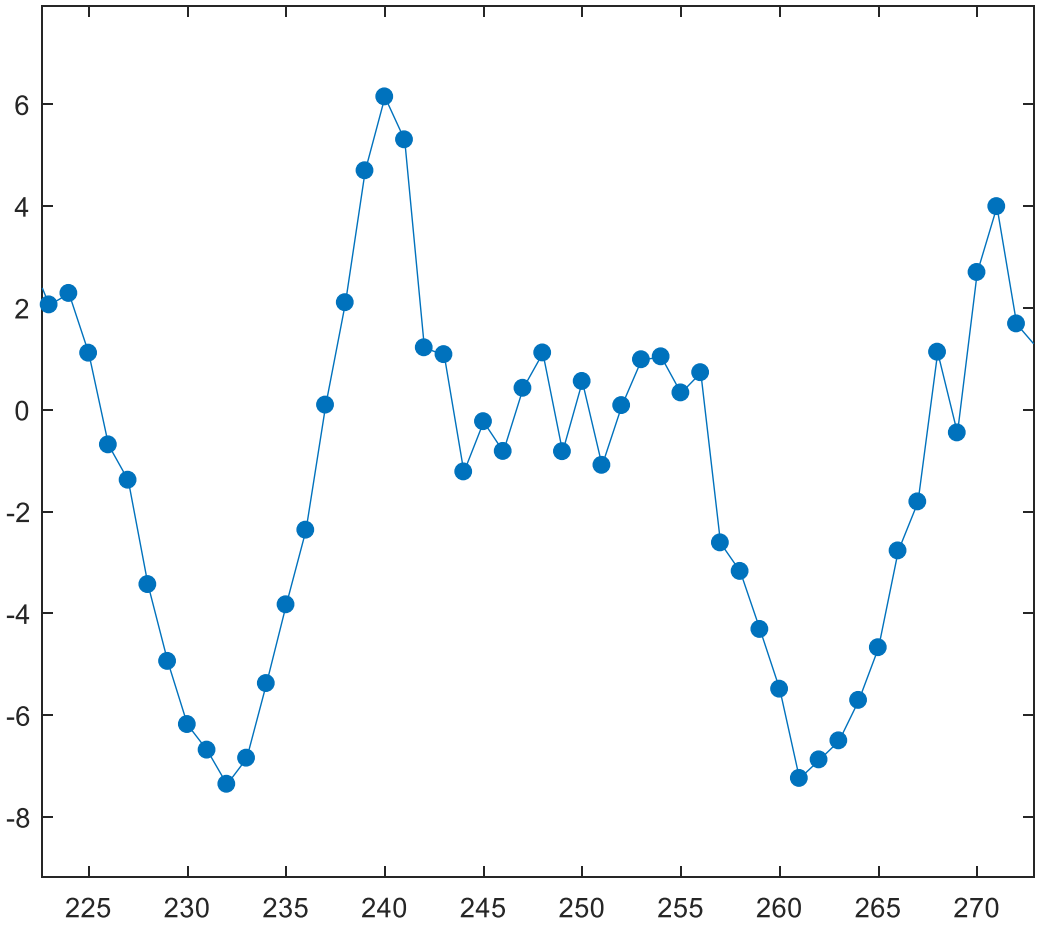
N = 30

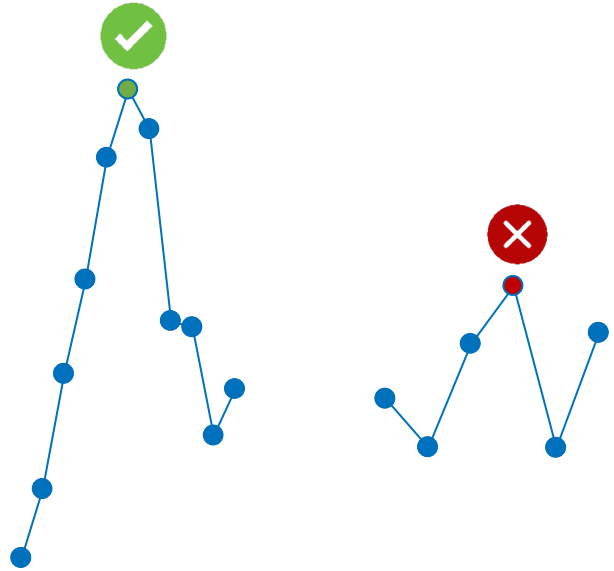
ONE WAY TO COUNT STEPS: DETECT ZERO CROSSINGS?



Smoothed signal.

ONE WAY TO COUNT STEPS: DETECT PEAKS?



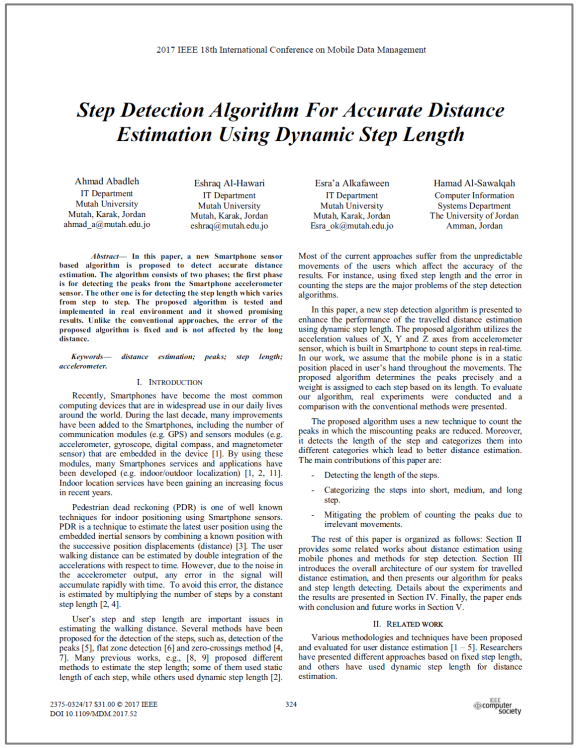


Final Think, Pair, Share

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



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.

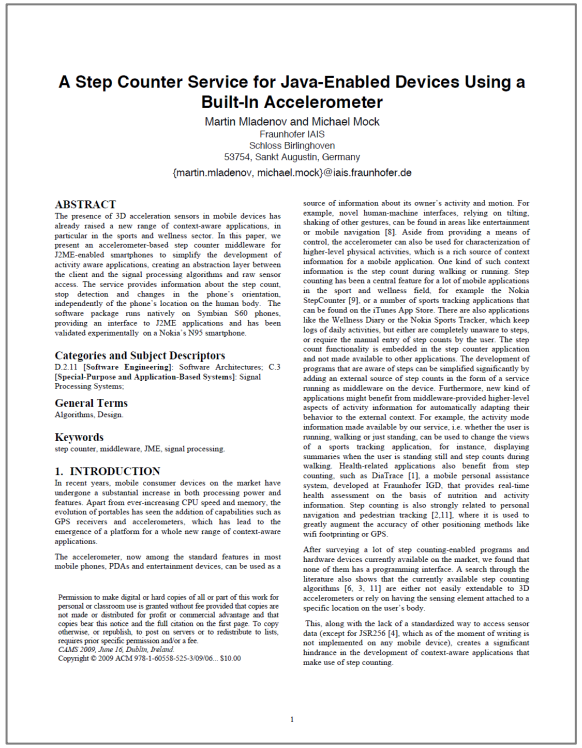
The rest of this paper is organized as follows: Section II provides some related works about distance estimation using mobile phones and methods for step detection. Section III introduces the overall architecture of our system for travelled distance estimation, and then presents our algorithm for peaks 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

324



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

1

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.*

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.

Fig. 1. Raw signals from accelerometer.

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

000213

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
Mladenov and Mock, CAMS'09

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?

SIGNAL PROCESSING: A UBICOMP APPROACH

Signal processing originated in the military with mathematical and electrical engineering roots (*e.g.,* sonar, radar). UbiComp is more applied.

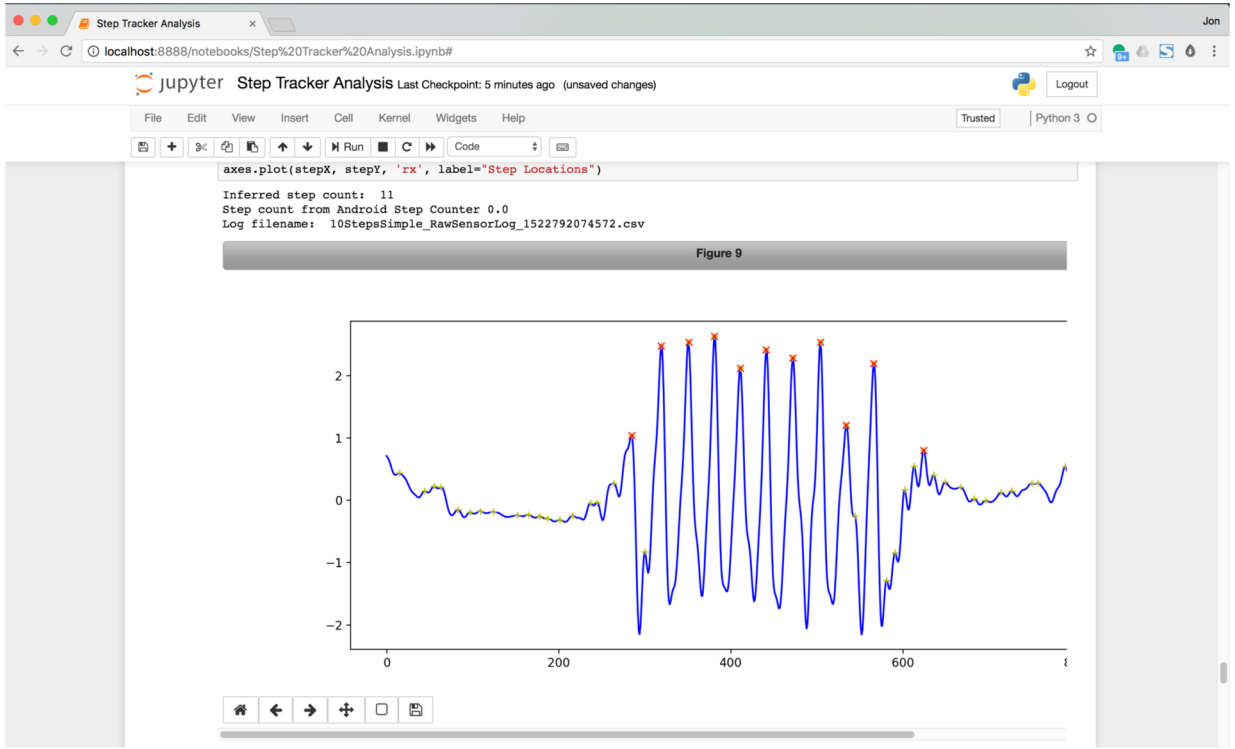
1. **Capture signals under controlled conditions** with "ground truth" (*e.g.,* walk ten steps, wait, walk ten steps, wait). Collect easy, medium, and hard examples.

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 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.



Jupyter Notebook

Numpy, scipy, matplotlib, reproducibility

SCHEDULE TODAY: 6:30-9:20

06:35-07:00: Discuss the readings

07:00-08:00: Signal Processing in UbiComp

08:00-08:10: Short break

08:10-09:20: Work on A1