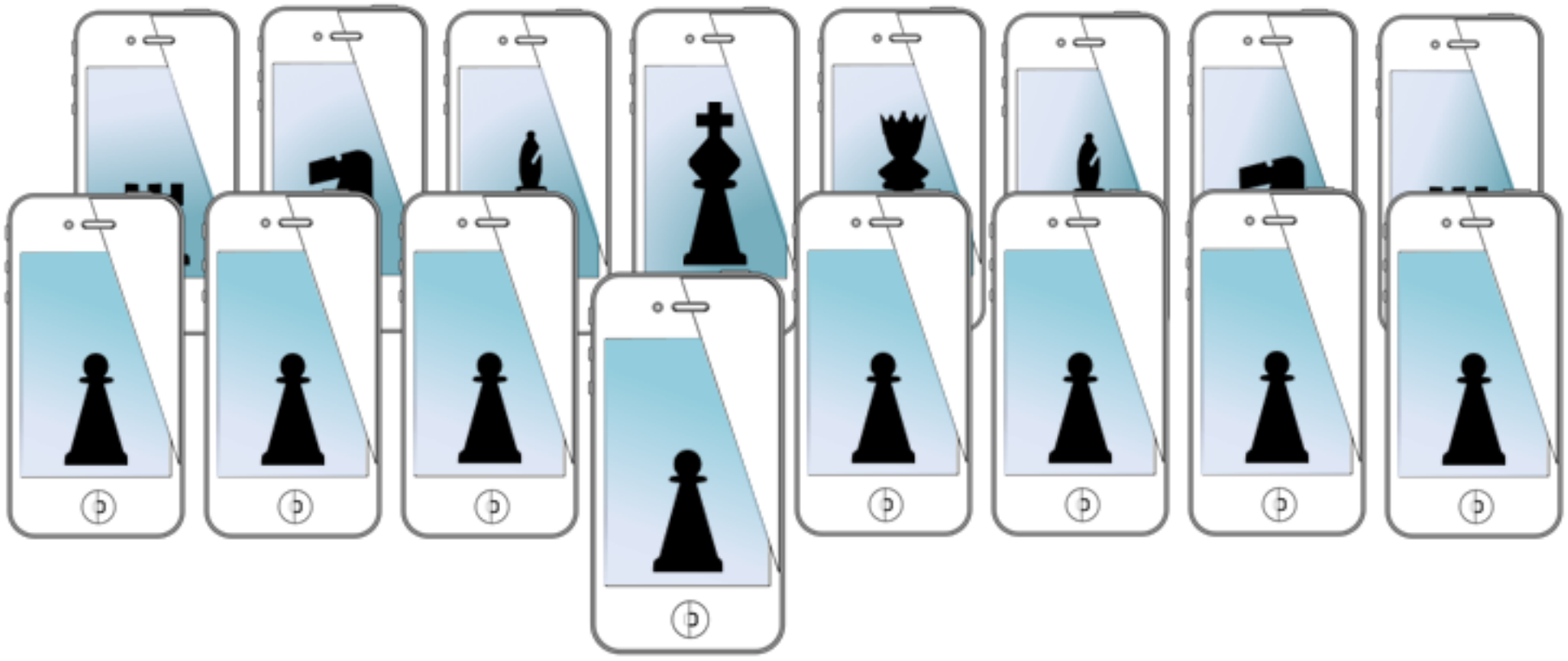
MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

doppler and activity monitoring

Eric C. Larson, Lyle School of Engineering,
Computer Science, Southern Methodist University

course logistics

- A1 grades update
- A2 is due soon!

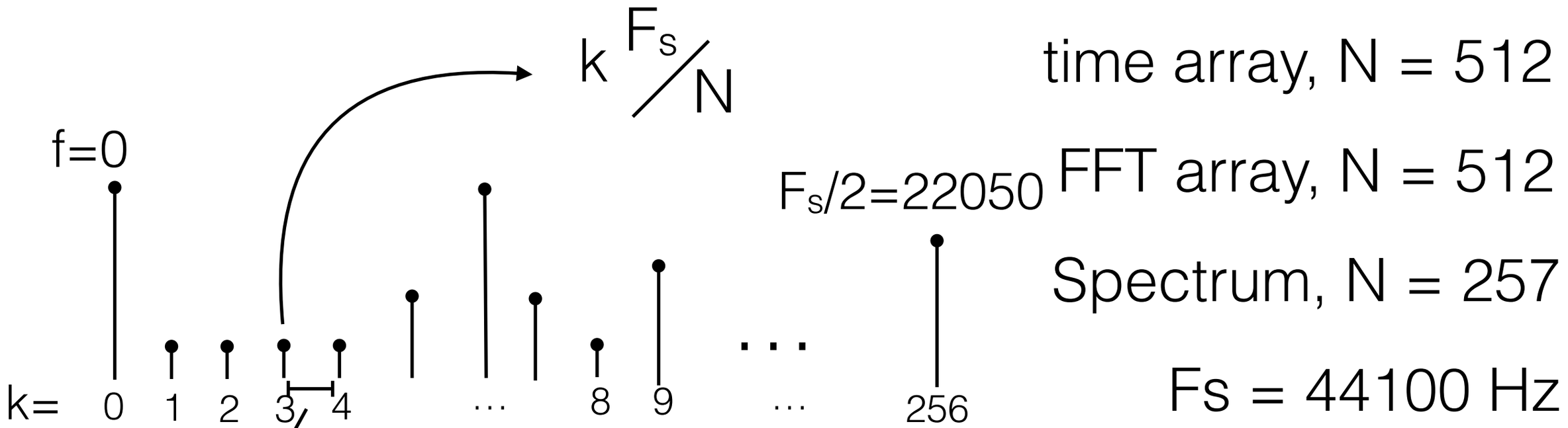
agenda

- general FFT review
- the doppler effect
- A2 explanations
- peak finding
- motion processing

FFT review

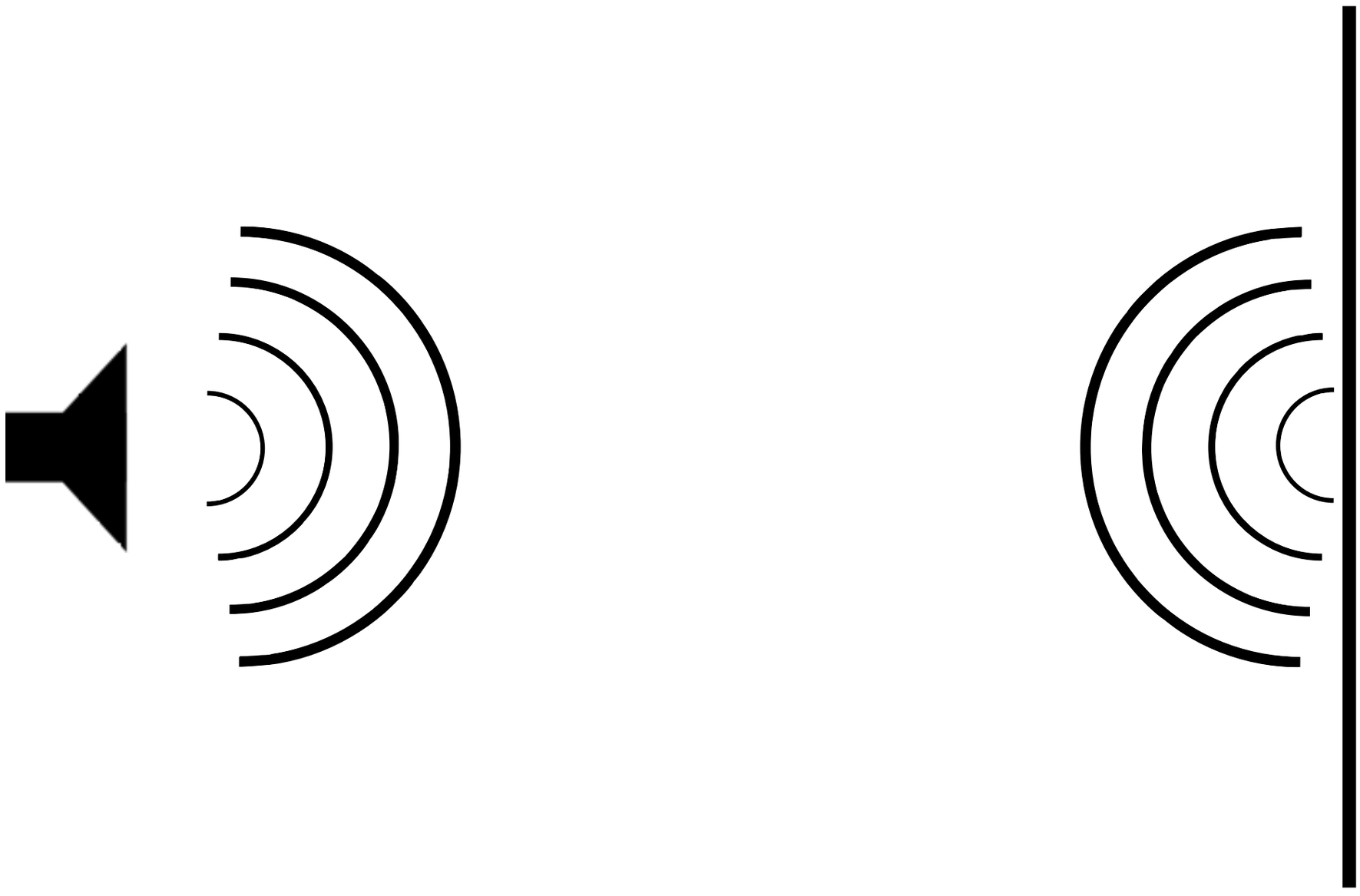
- sampling rate
  - dictates the time between each sample, (1 / sampling rate)
  - max frequency we can measure is half of sampling rate
- resolution in frequency
  - tradeoff between length of FFT and sampling rate
- each frequency “bin” is an index in the FFT array
  - each bin represents (Fs / N) Hz
  - what does that mean for 6 Hz accuracy?

time and frequency

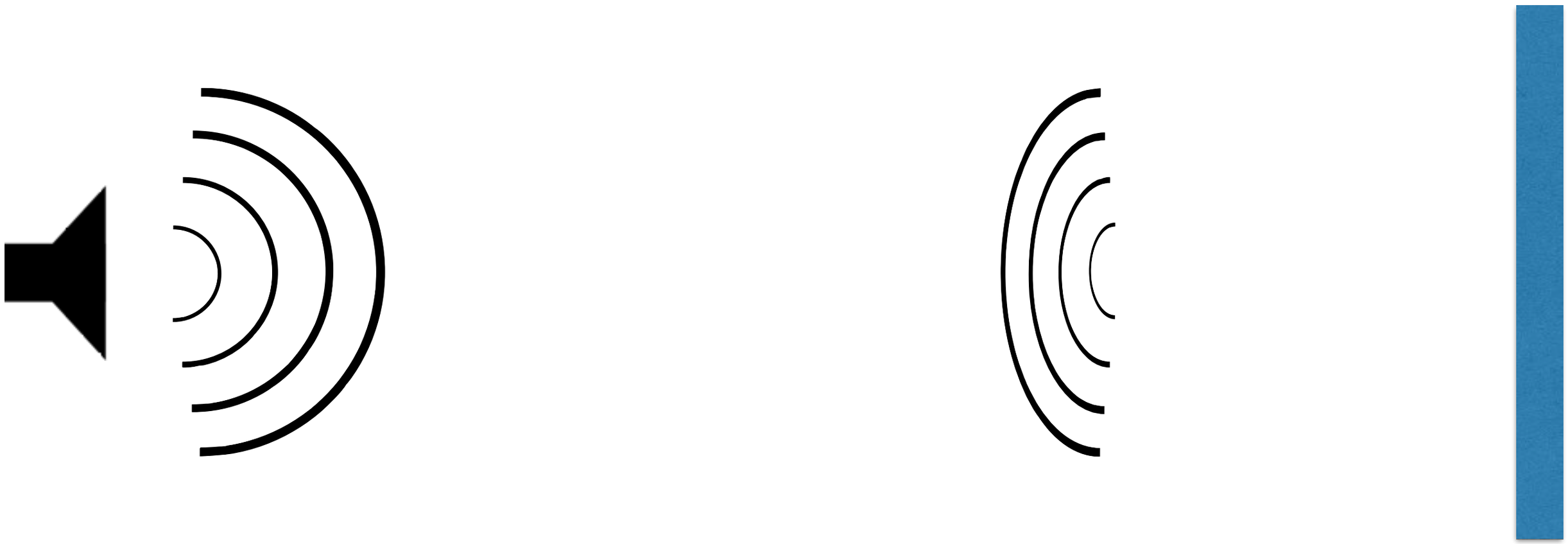


$$df = \frac{F_s/2}{N/2} = F_s/N = 44100/512 = \sim 86 \text{ Hz}$$

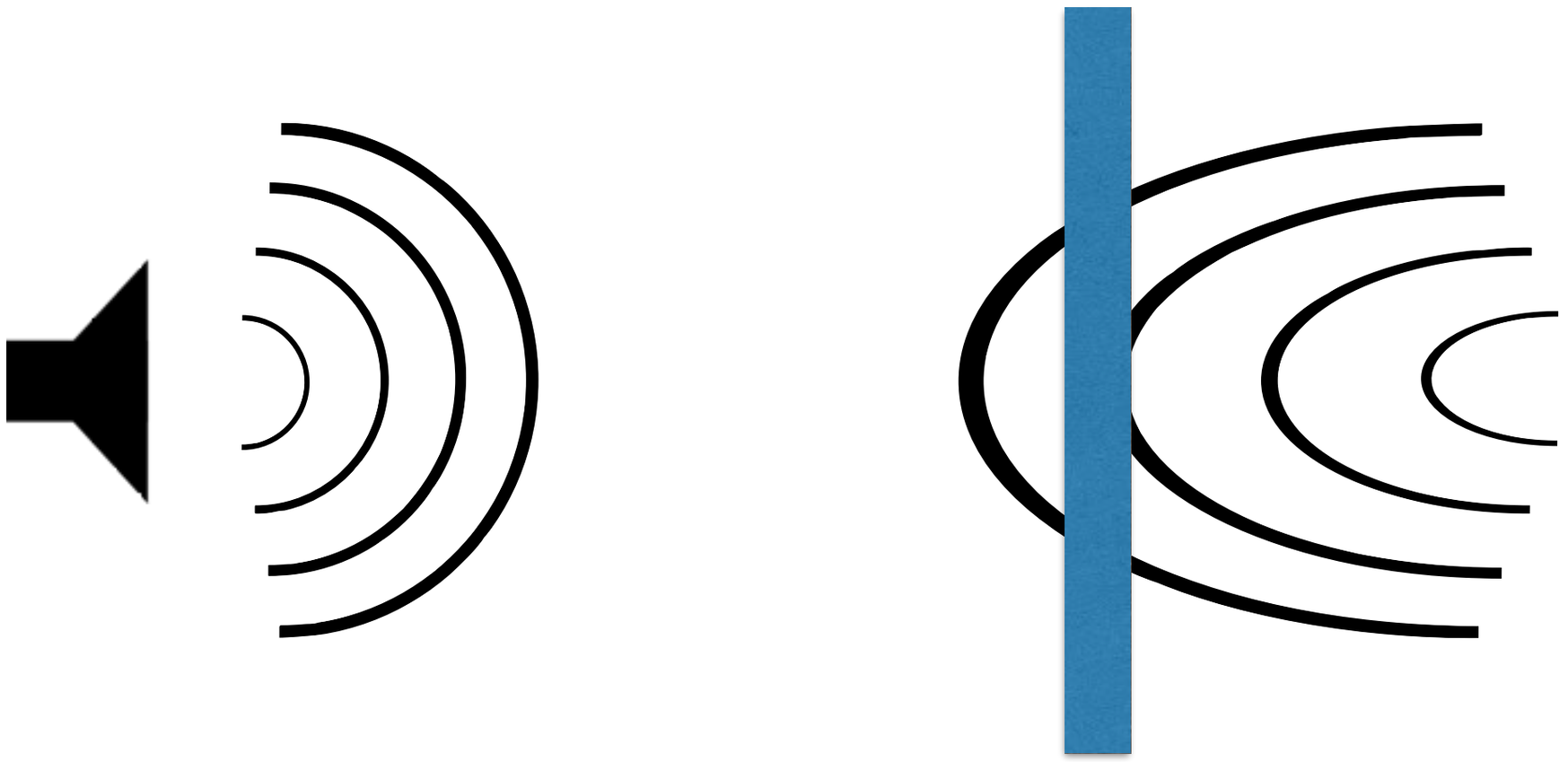
the doppler effect



the doppler effect



the doppler effect



the doppler effect

$$\Delta f = \frac{V_{object}}{c} f_0$$

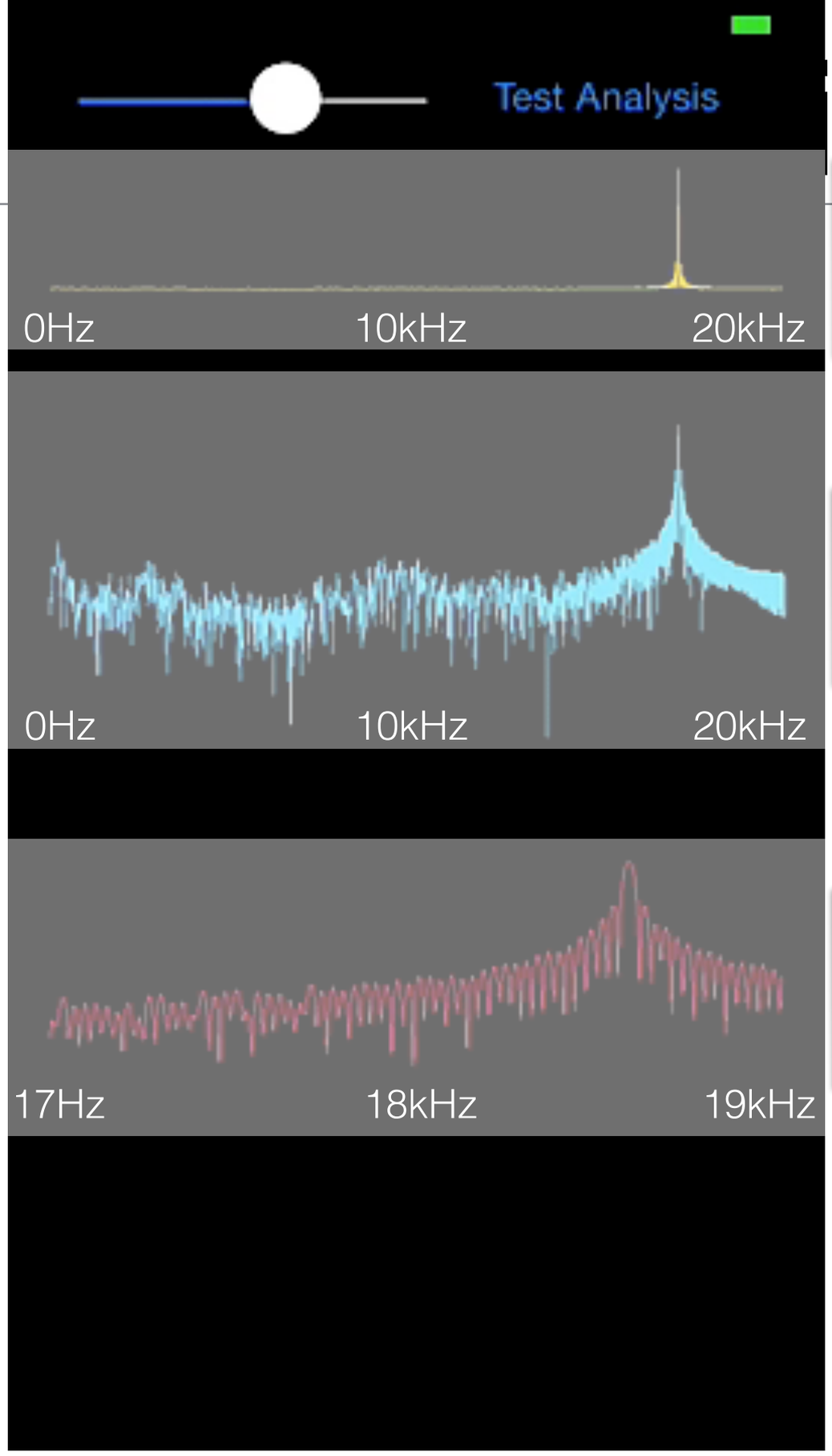
change in frequency

velocity of object

speed of sound

frequency of source

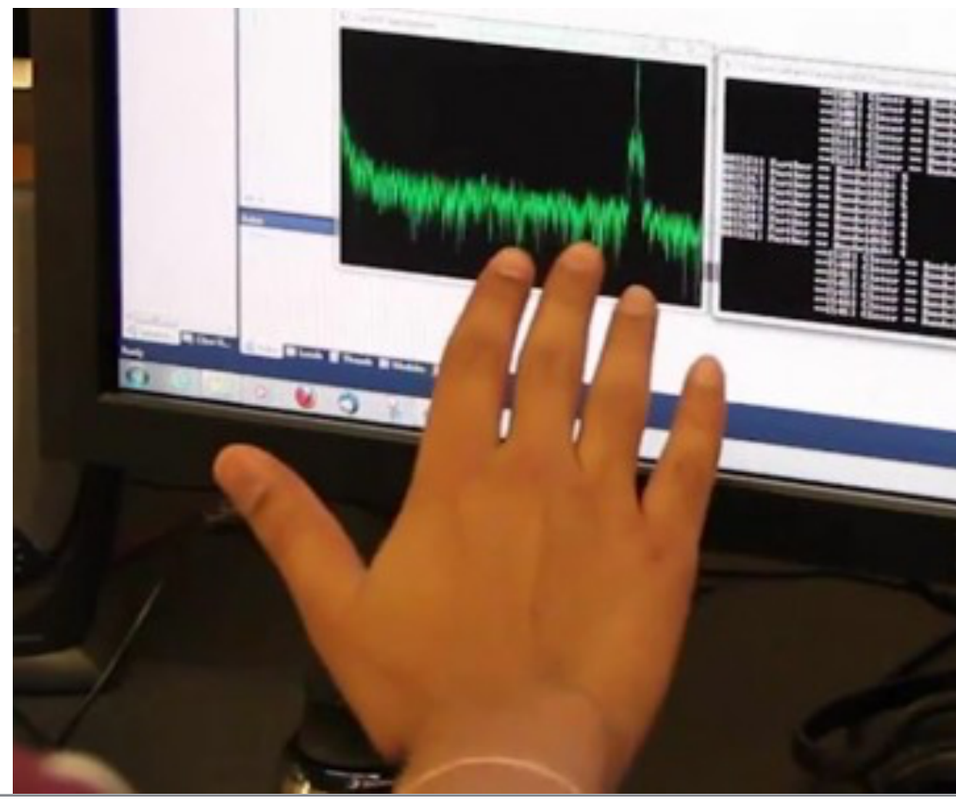
ifts fron



linear

db

db zoomed (freq axis)

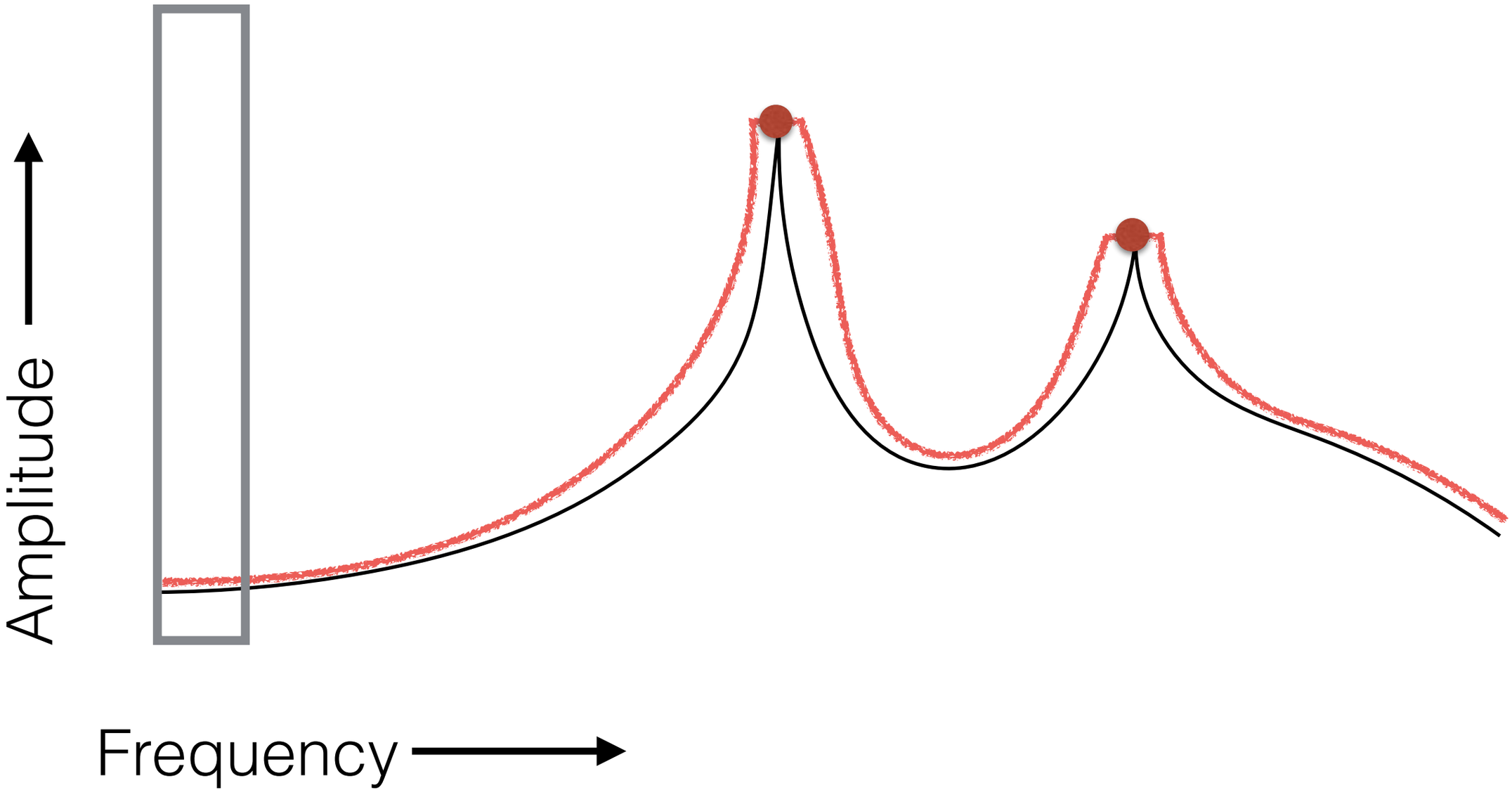


A2 specifications

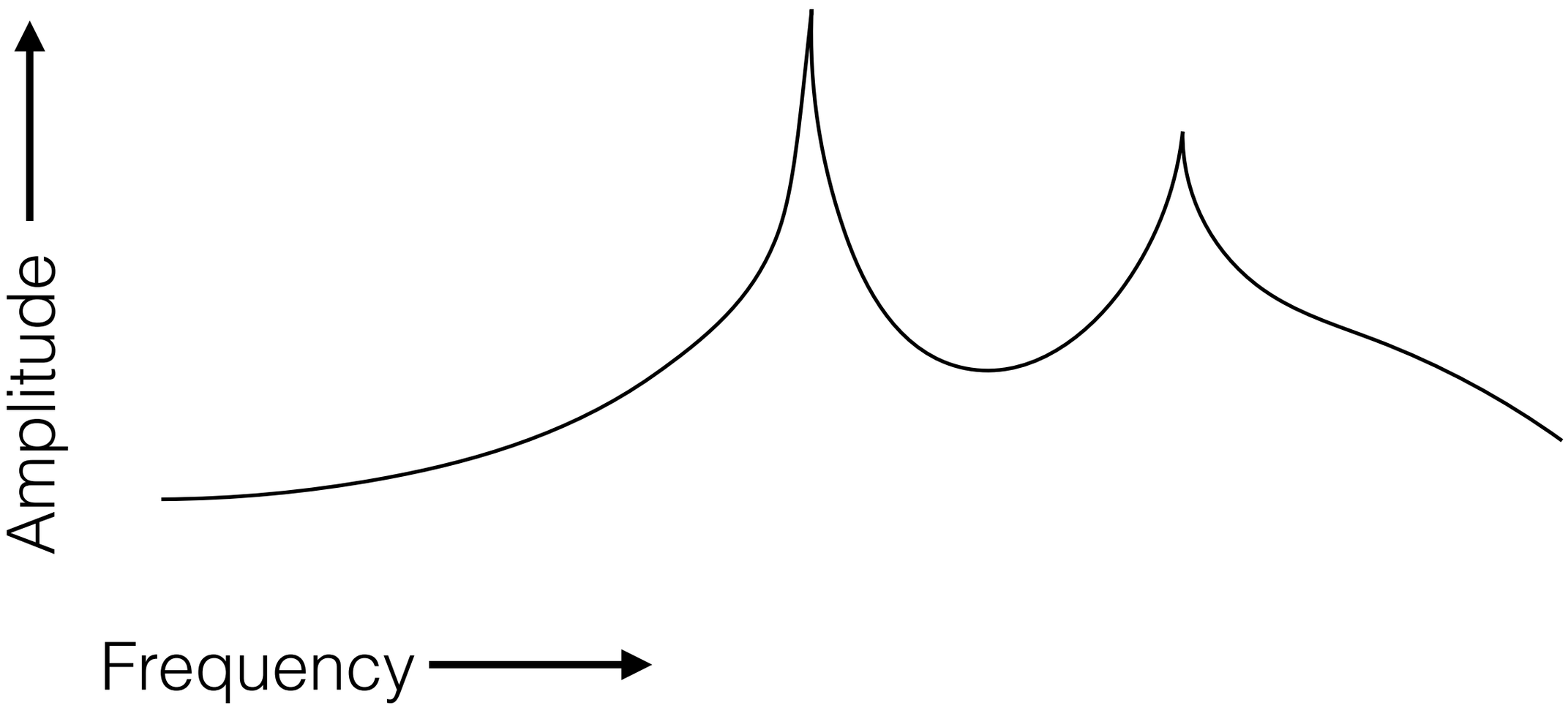
On Canvas

local peak finding

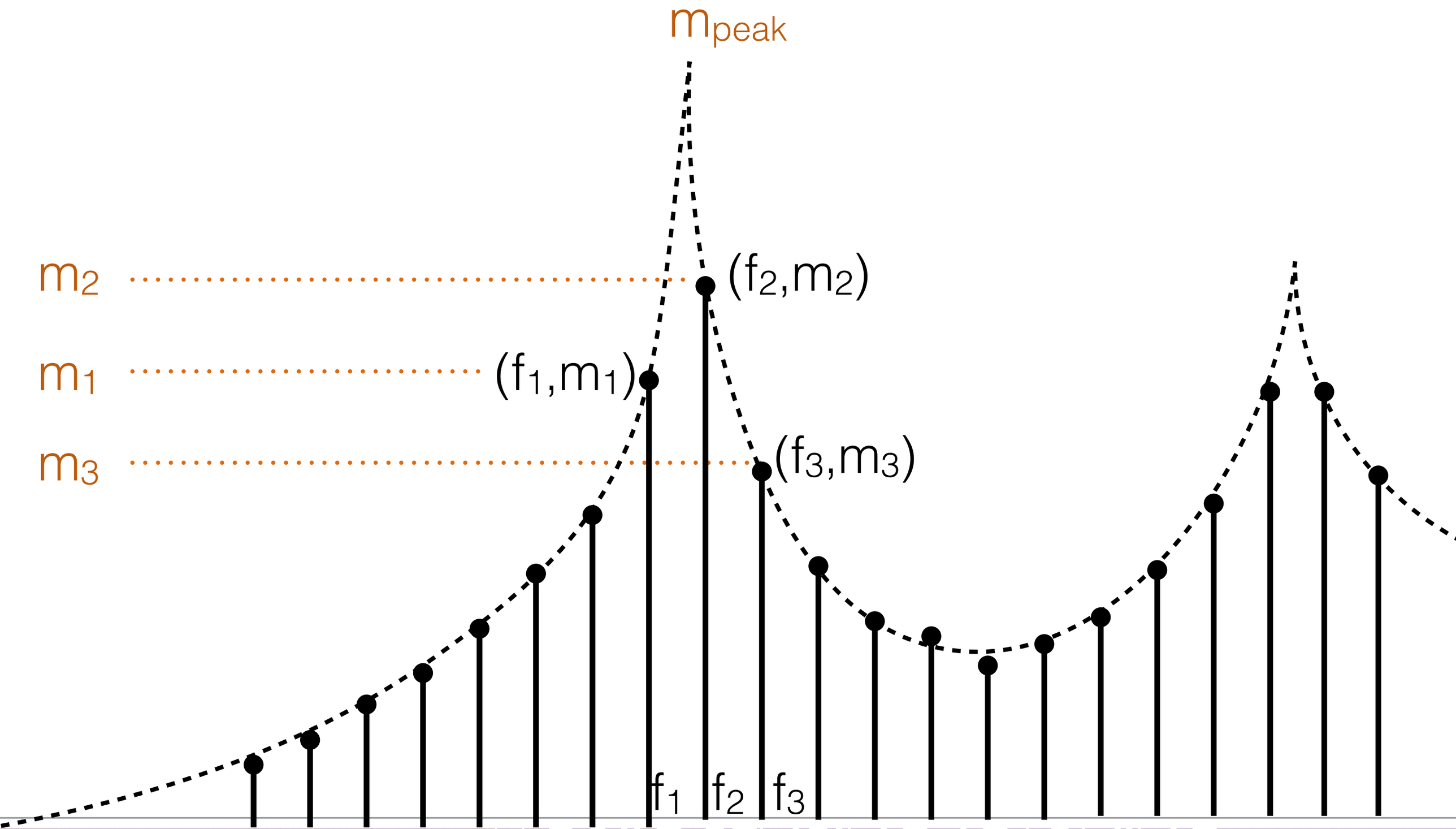
max in window



peak interpolation

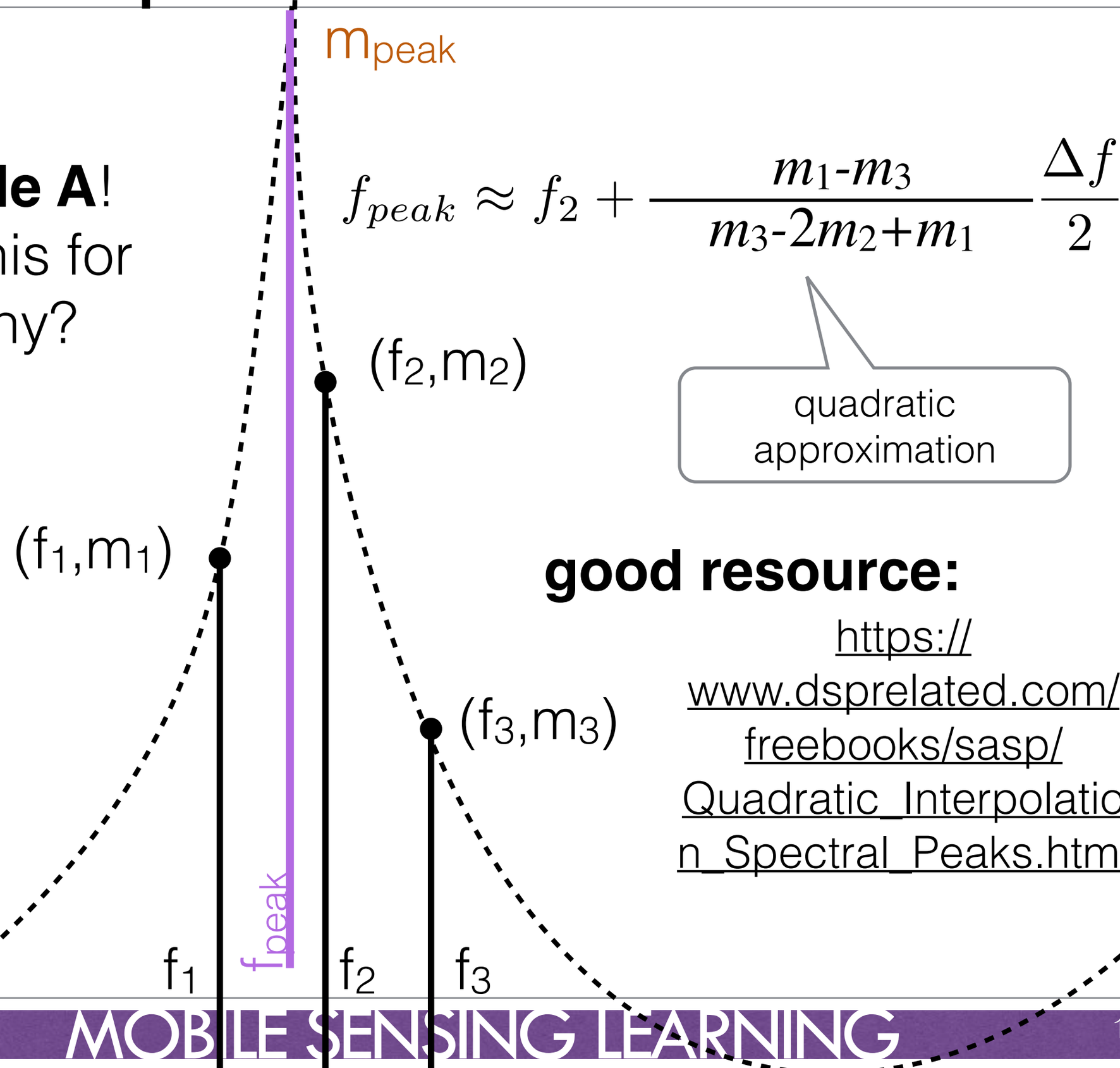


peak interpolation



peak interpolation

great for **module A**!
no need to do this for
module B, Why?



$$f_{peak} \approx f_2 + \frac{m_1\text{-}m_3}{m_3\text{-}2m_2\text{+}m_1}\frac{\Delta f}{2}$$

quadratic approximation

good resource:

https://www.dsprelated.com/freebooks/sasp/Quadratic_Interpolation_Spectral_Peaks.html

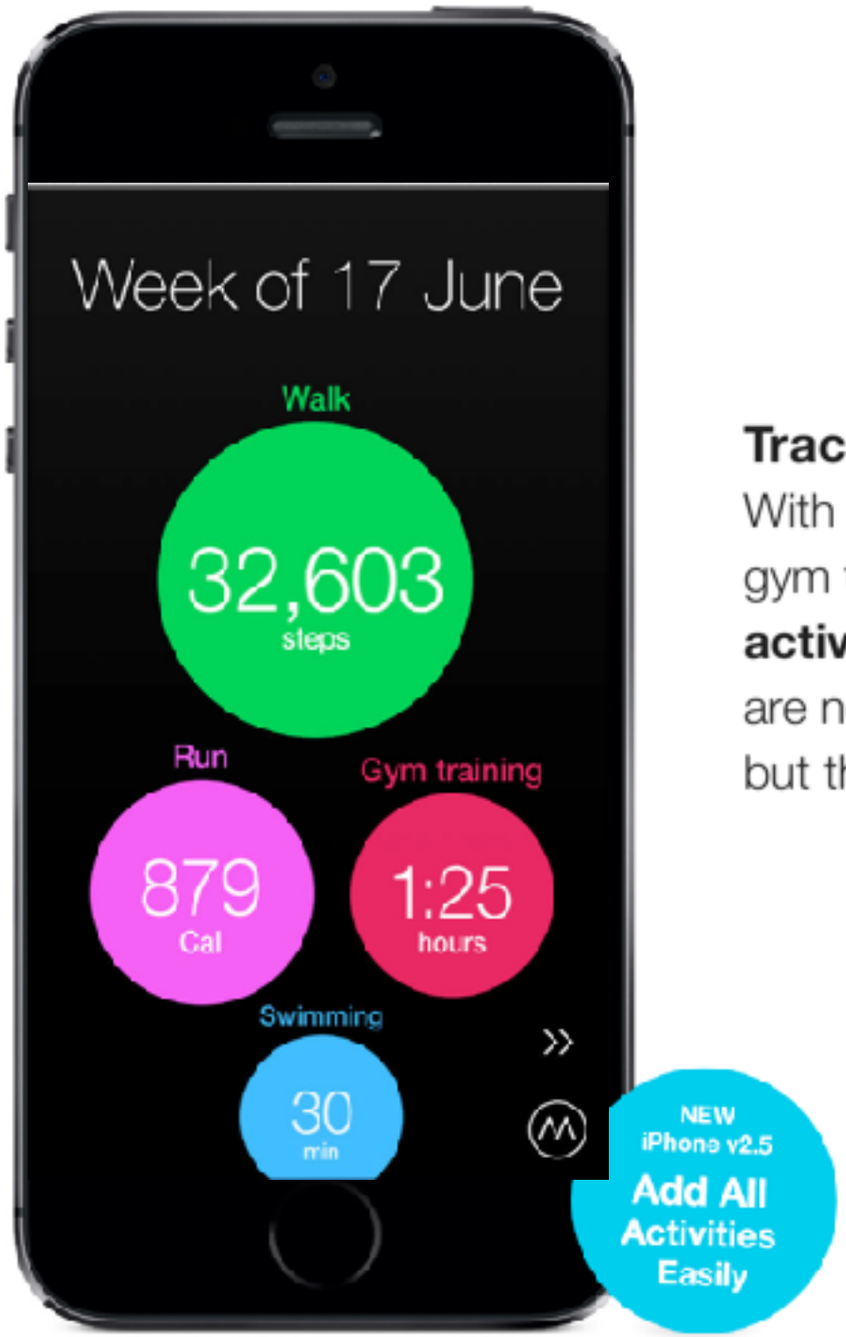
Questions on the FFT/audio

- we are about to move to motion processing…
- so ask now!
- …or later…

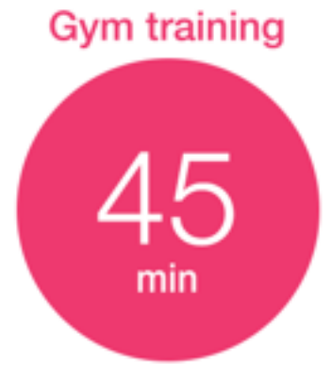
and now…

- no more microphone data!
- let's get data from the other sensors…
- core motion
  - the M# co-processor

a nice example of core motion



Track all activity*
With Moves 2.5 for iPhone, you can add gym training and **over 60 other activities** by duration. These activities are not (yet!) automatically recognized, but they are easy to add.

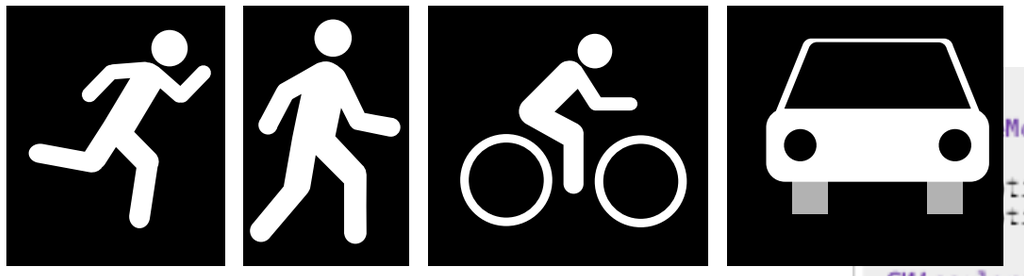


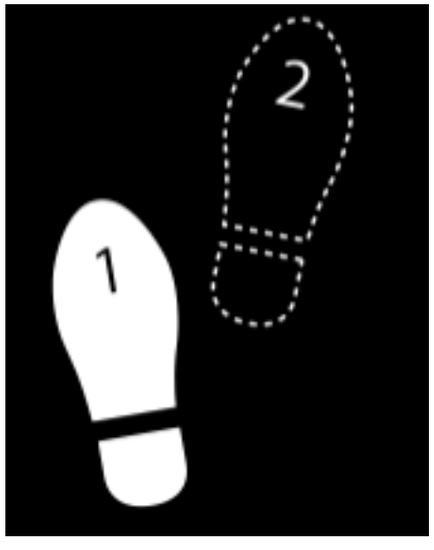
the M- coprocessor

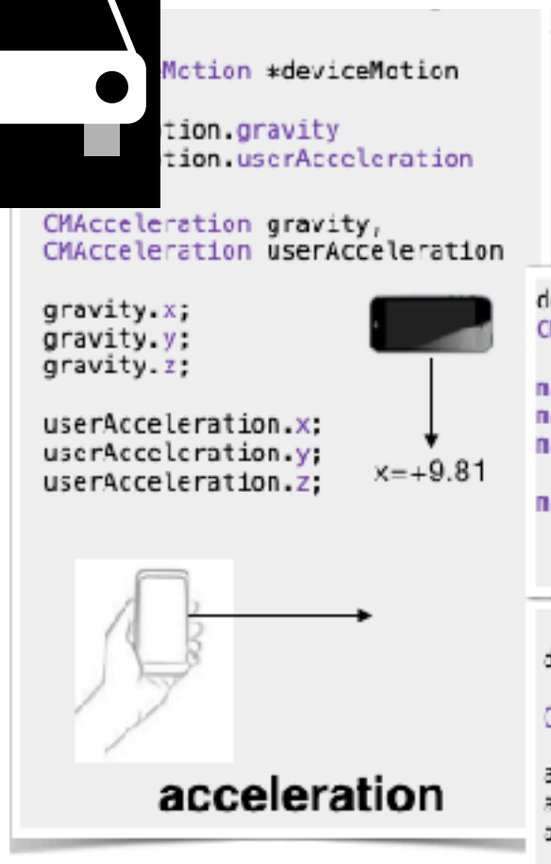
- 150MHz+ processor that reads all motion data from all “motion” sensors on the phone
    - accelerometer
    - magnetometer (compass)
    - gyroscope
    - barometer! (M8 and above)
- mediates all access to data
    - battery life++
    - parallel processing++
    - overhead += 0, seriously
- sensor fusion for more accurate analysis, very cool

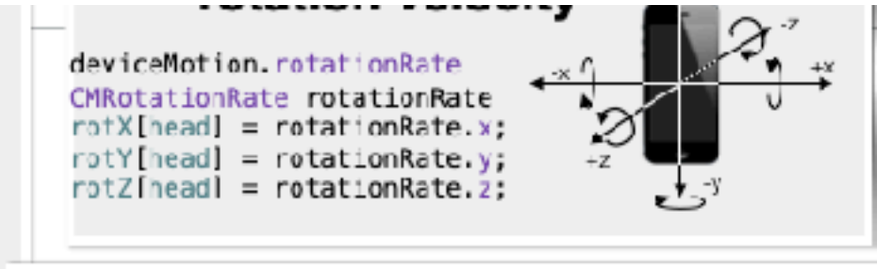
motion lecture agenda

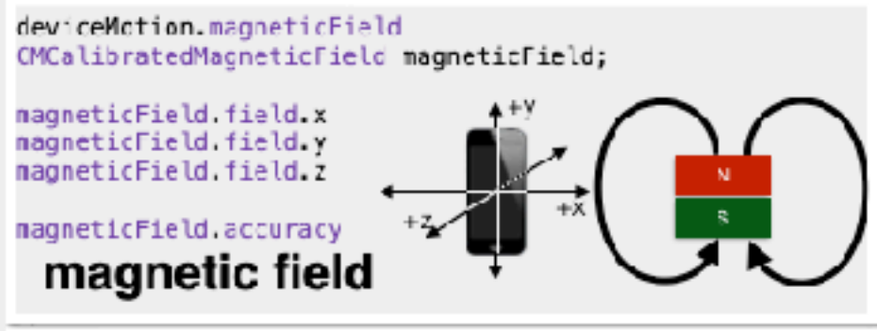
- today: activity recognition through API
- today: pedometer step counting through API
- next time: raw motion data gathering

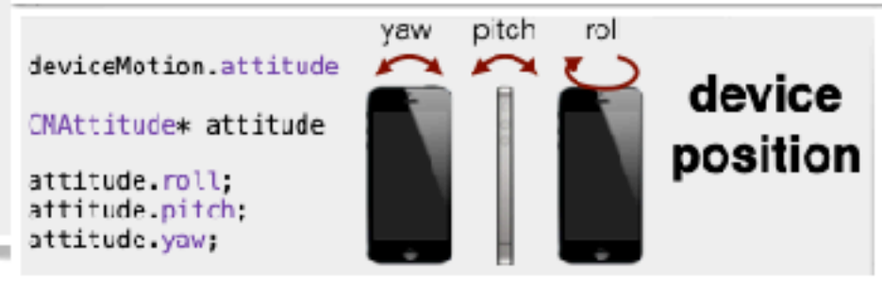






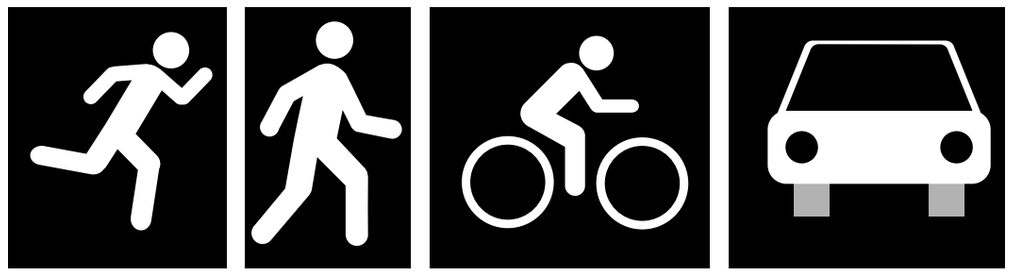






high level streams

- not just raw data!
  - the M- co-processor does sophisticated analysis of sensor data for you
  - enables naive access to “high level” information
- can register your app to receive “updates” from the co-processor unit
  - steps taken (and saved state of steps)
  - some common activity
    - running, walking, **cycling**, still, in car, unknown



activity from M-

- uses the “core motion” framework (CM)
- mediated through the “CMActivityManager”
  - is device capable of activity?
  - query past activities (up to 7 days)
  - subscribe to changes
- interaction completely based on blocks and handlers

More help: https://developer.apple.com/videos/wwdc/2014/
Navigate to: **Motion Tracking and Core Motion Framework**

subscribing to activity

- updates are notifications

```
#import <CoreMotion/CoreMotion.h>

// from co-processor
@property (nonatomic, strong) CMMotionActivityManager *motionActivityManager;

   // initialize the activity manager (check if available)
    if ([CMMotionActivityManager isActivityAvailable] == YES) {
        self.motionActivityManager = [[CMMotionActivityManager alloc] init];
    }

    [self.motionActivityManager startActivityUpdatesToQueue:[NSOperationQueue mainQueue]
                                                 withHandler:^(CMMotionActivity *activity) {
                                                         // do something with the activity info!
                                                 }];
     NSLog(@"Activity Manager Running");

    [self.motionActivityManager stopActivityUpdates];
```

import framework

declare activity manager

device capable?

instantiate

subscribe

queue to run on

block to handle updates

end subscription

swift version

```
import CoreMotion


let activityManager = CMMotionActivityManager()
let customQueue = OperationQueue() // not the main

 override func viewDidLoad() {
     super.viewDidLoad()

     if CMMotionActivityManager.isActivityAvailable(){
         self.activityManager.startActivityUpdatesToQueue(customQueue)
         { (activity:CMMotionActivity?)  -> Void in
             NSLog("%@",activity!.description)
         }
     }
 }

 override func viewWillDisappear(animated: Bool) {
     if CMMotionActivityManager.isActivityAvailable() {
         self.activityManager.stopActivityUpdates()
     }
     super.viewWillDisappear(animated)
 }
```

import framework

declare activity manager

device capable?

closure to handle updates
(*this one just prints description*)

end subscription

what's in an update?

- updated when any part of activity estimate changes
- each update is a CMMotionActivity class instance
  - startDate (down to seconds)
  - walking {0,1}
  - stationary {0,1}
  - running {0,1}
  - cycling {0, 1}
  - automotive {0,1}
  - unknown {0,1}
  - confidence {Low, Medium, High}

```
startActivityUpdatesToQueue:[NSOperationQueue mainQueue]
                withHandler:^(CMMotionActivity *activity)
{
    // do something with the activity info!
                                        }];
```

```
self.activityManager.startActivityUpdatesToQueue(customQueue)
                { (activity:CMMotionActivity?)  -> Void in
                // do something with the activity info!
            }
```

example update

inside handler

```
 startActivityUpdatesToQueue:[NSOperationQueue mainQueue]
                 withHandler:^(CMMotionActivity *activity) {
                // do something with the activity info!
                                        }];

  // enum for confidence is 0=low,1=medium,2=high
NSLog(@" confidence:%ld \n stationary: %d \n walking: %d \n run: %d \n cycle %d \n in car: %d",
          activity.confidence,
          activity.stationary,
          activity.walking,
          activity.running,
          activity.cycling,
          activity.automotive);


    switch (activity.confidence) {
        case CMMotionActivityConfidenceLow:
            self.confidenceLabel.text = @"low";
            break;
        case CMMotionActivityConfidenceMedium:
            self.confidenceLabel.text = @"med.";
            break;
        case CMMotionActivityConfidenceHigh:
            self.confidenceLabel.text = @"high";
            break;
        default:
            break;
    }
```

from notification

access fields easily

look at confidence

what's in an update?

Example Scenarios

Device scenarios	stationary	walking	running	automotive	cycling	unknown
On table	true	false	false	false	false	false
On runner's upper arm	false	false	true	false	false	false
In dash of idling vehicle	true	false	false	true	false	false
In dash of moving vehicle	false	false	false	true	false	false
Passenger checking email	false	false	false	false	false	false
Immediately after reboot	false	false	false	false	false	true
In zumba class	false	false	false	false	false	false

https://developer.apple.com/videos/wwdc/2014/

past activity

- query for an array of CMMotionActivity activities

```
// example of querying from certain dates
NSDate *now  = [NSDate date];
NSDate *from = [NSDate dateWithTimeInterval:-60*60*24 sinceDate:now];



[self.motionActivityManager queryActivityStartingFromDate:from
                toDate:now
                toQueue:[NSOperationQueue mainQueue]
  withHandler:^(NSArray *activities, NSError *error) {

    for(CMMotionActivity *cmAct in activities)
    {
        NSLog(@"At %@, user was walking %d",cmAct.startDate,cmAct.walking);
    }

}];
```

setup date range

set dates

set queue

handle error!

handle output

- can you guess what the swift code looks like?

what's in an update?

Motion Activity

Walking

Performance is fairly insensitive to location

- Detection can be suppressed when device is in hand

Relatively low latency

Very accurate, on average

- Expect intermittent transitions into and out of walking state

https://developer.apple.com/videos/wwdc/2014/

what's in an update?

Motion Activity

Running

Completely insensitive to location

Shortest latency

Most accurate classification

https://developer.apple.com/videos/wwdc/2014/

what's in an update?

Motion Activity

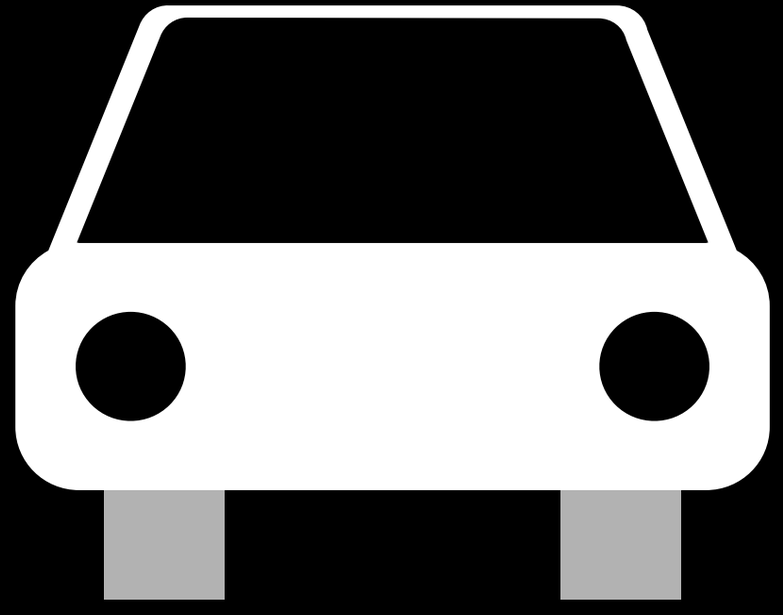
Automotive

Performance is sensitive to location

- Works best if device is mounted, or placed in dash or in cup holder

Variable latency

Relies on other information sources when available



https://developer.apple.com/videos/wwdc/2014/

what's in an update?

Motion Activity

Cycling

Performance is very sensitive to location

- Works best if device is worn on upper arm

Longest latency

- Best for retrospective use cases

https://developer.apple.com/videos/wwdc/2014/

Motion Processing Architecture

real time subscription

To clients

query past data

To clients

Activity Classifier

T=~2.5s

Filter

Additional info sources

https://developer.apple.com/videos/wwdc/2014/

more than activity

- M- also tracks pedometer information during each activity
- like activity: setup as a **push** system (subscribe)

- pedometer: special handling from the M?
  - CMStepCounter is deprecated as of 2015!!!!!!! Do not use it…
  - instead, we will use the CMPedometer class

pedometer use

declare and init

available on this device?

```
let pedometer = CMPedometer()

if CMPedometer.isStepCountingAvailable(){
    pedometer.startPedometerUpdatesFromDate(NSDate())
        { (pedData: CMPedometerData?, error:NSError?) -> Void in
            NSLog("%@",pedData.description)
        }
}

if CMPedometer.isStepCountingAvailable(){
    self.pedometer.stopPedometerUpdates()
```

closure handler for updates

unsubscribe

pedometer

- do not rely on the update to:
  - have any regularity
  - be what you asked for
- iOS: you get the update when we say you do!
  - which optimizes battery life
  - is not at expense of interaction
  - minimizes bus traffic on co-processor
  - is more accurate
  - and will keep track even if your app is in the background

pedometer use

revisiting

declare and init

available on this device?

properties from step counter

```
let pedometer = CMPedometer()

if CMPedometer.isStepCountingAvailable(){
    pedometer.startPedometerUpdatesFromDate(NSDate())
        { (pedData: CMPedometerData?, error:NSError?) -> Void in
            NSLog("%@",pedData.description)
        }
}
```

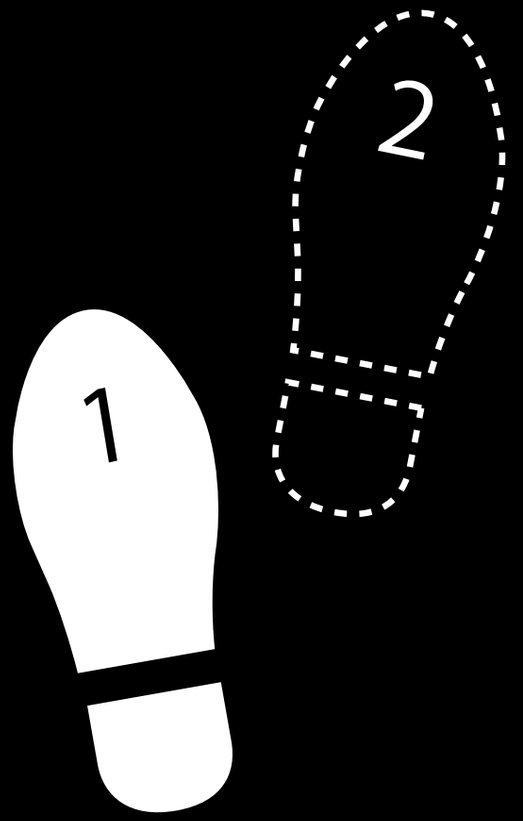
```
if CMPedometer.isStepCountingAvailable(){
    self.pedometer.stopPedometerUpdates()
```

unsubscribe

```
CMPedometerData,<startDate 2021-09-21
13:56:54 +0000 endDate 2021-09-21 13:57:17
+0000 steps 35 distance 27.57728308765218
floorsAscended 0 floorsDescended 0
currentPace 0.5944125511973894
currentCadence 2.172188043594 36
averageActivePace 0.6163431784950018>
```

Pedometer

Step counting



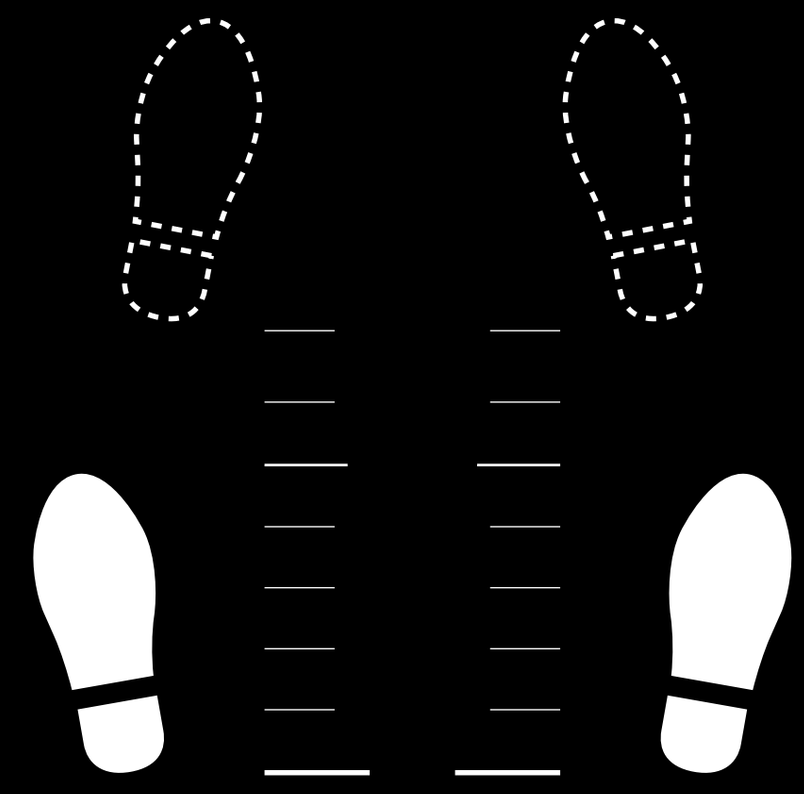
Consistent performance across body locations

Extremely accurate

Robust to extraneous motions

Pedometer

Stride estimation



Consistent performance across body locations

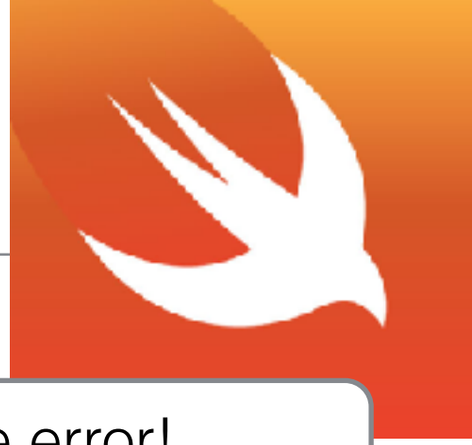
Consistent performance across pace

Extremely accurate

Adapts to the user over time

https://developer.apple.com/videos/wwdc/2014/

querying past steps



handle error!

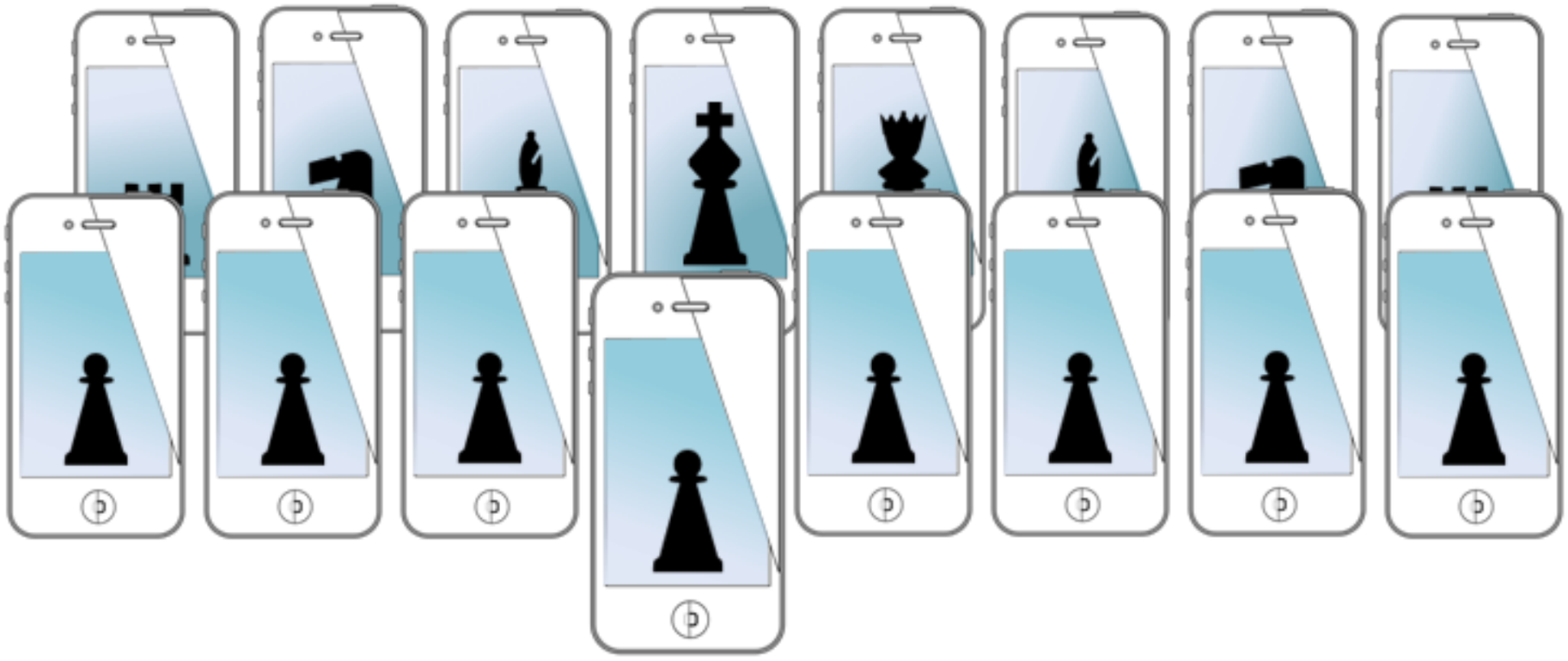
```
let now = NSDate()
let from = now.dateByAddingTimeInterval(-60*60*24)

self.pedometer.queryPedometerDataFromDate(from, toDate: now)
{ (pedData: CMPedometerData?, error: NSError?) -> Void in

    let aggregated_string = "Steps: \(pedData.numberOfSteps) \n
            Distance \(pedData.distance) \n
            Floors: \(pedData.floorsAscended.integerValue)"

    dispatch_async(dispatch_get_main_queue()){
        self.activityLabel.text = aggregated_string
    }
}
```

access properties

M- pedometer/activity demo

if time!

MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

doppler and activity monitoring

Eric C. Larson, Lyle School of Engineering,
Computer Science, Southern Methodist University