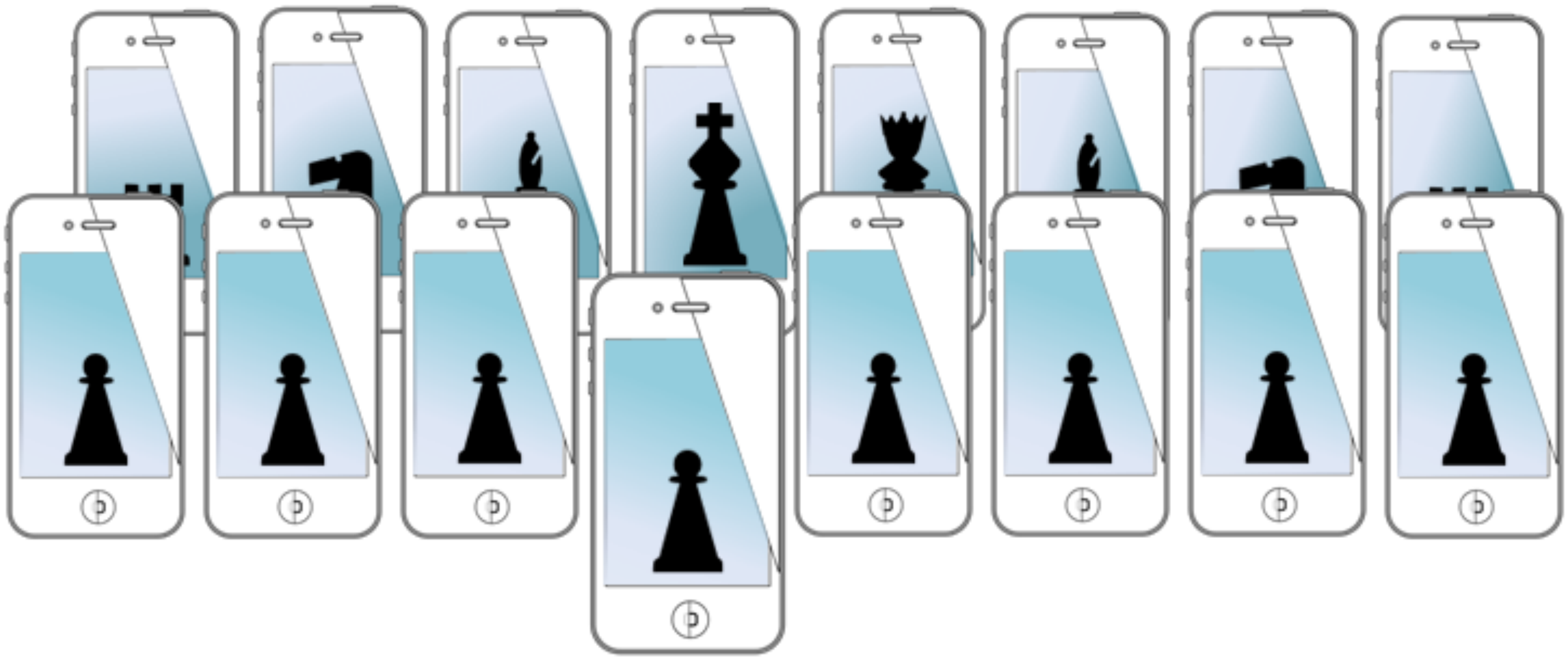
MOBILE SENSING LEARNING



CSE5323 & 7323

Mobile Sensing and Learning

week 5, lecture a: doppler and activity monitoring

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

course logistics

- A1 grades coming soon
- A2 is due Friday

agenda

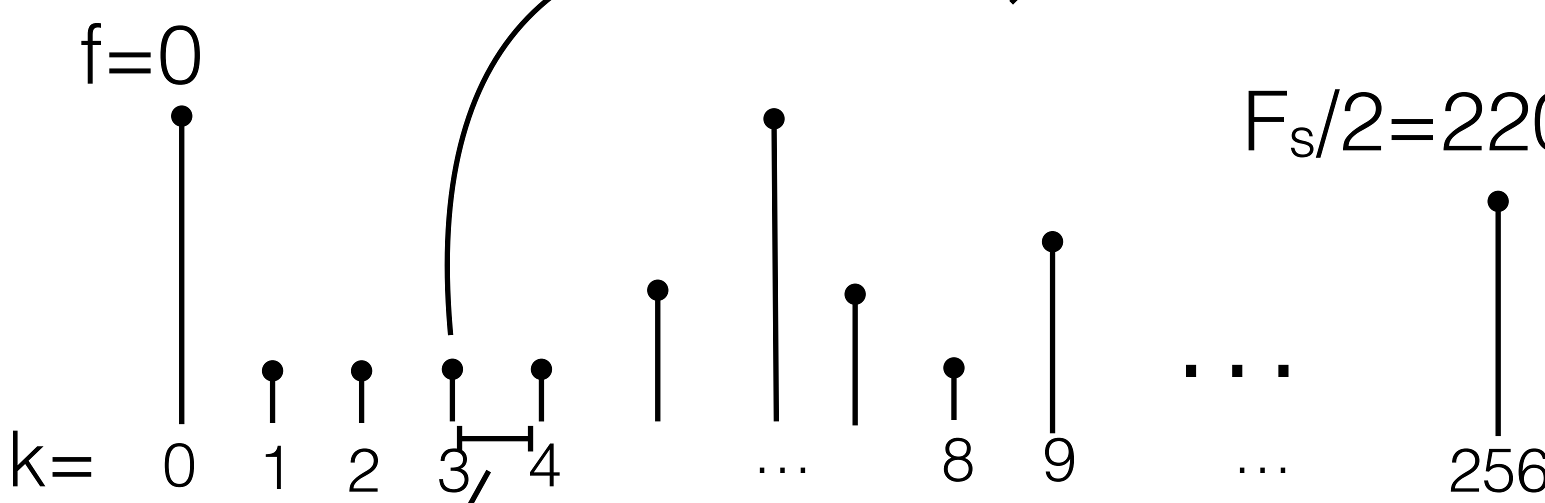
- general FFT review
- the doppler effect
- A2 explanations
- peak finding
- motion processing

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

$$k\,{}^{F_s}\!/_{N}$$



time array, N = 512

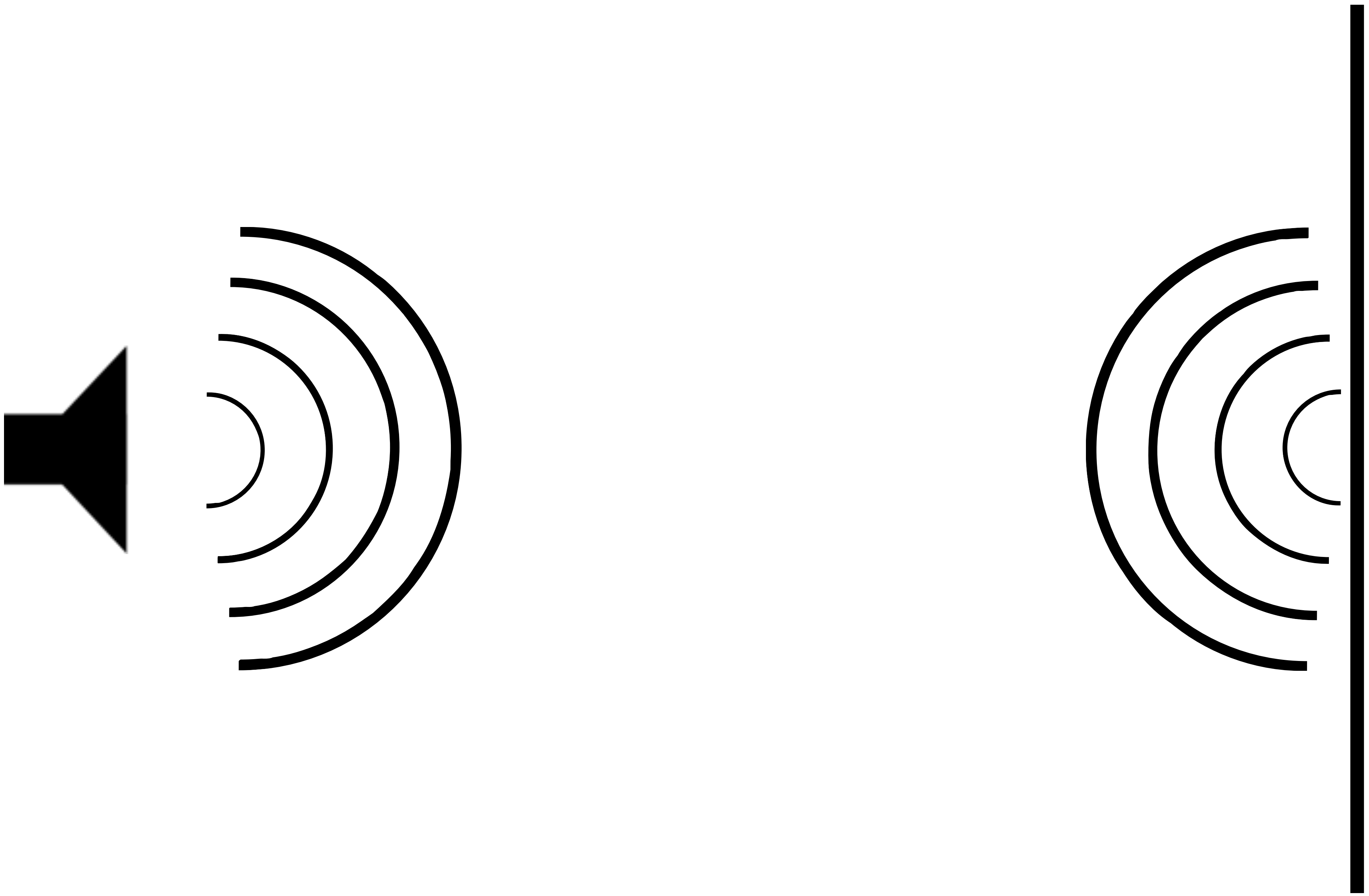
FFT array, N = 512

Spectrum, N = 257

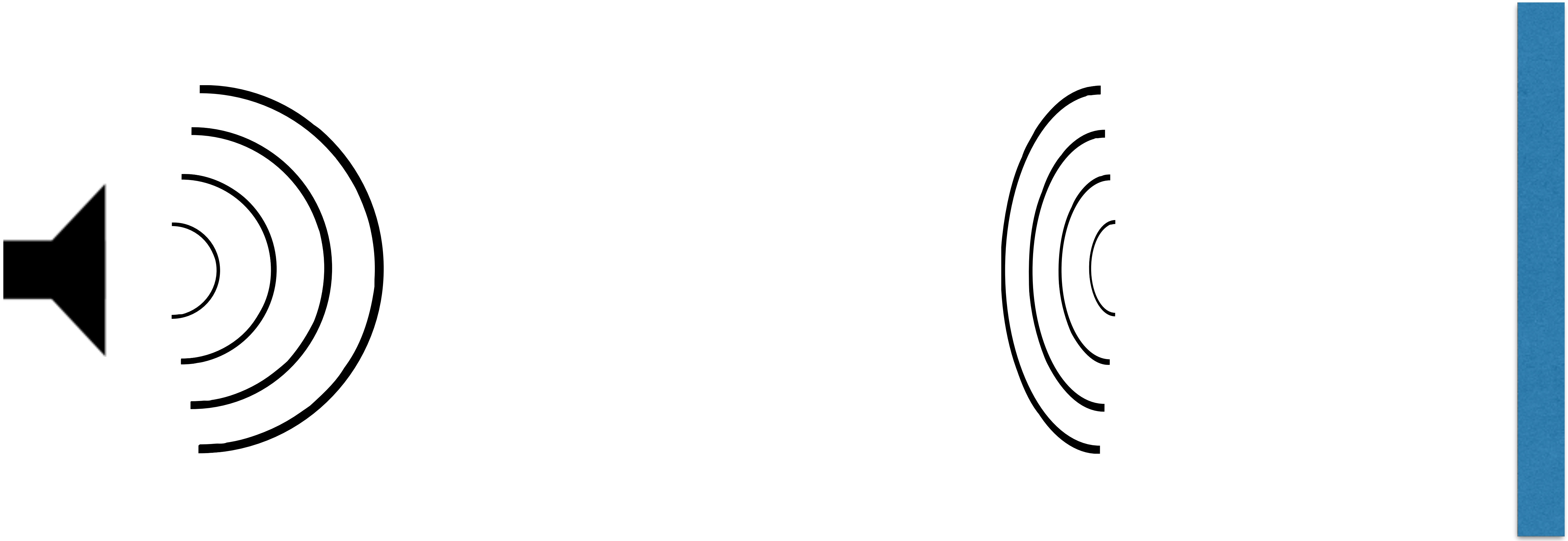
Fs = 44100 Hz

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

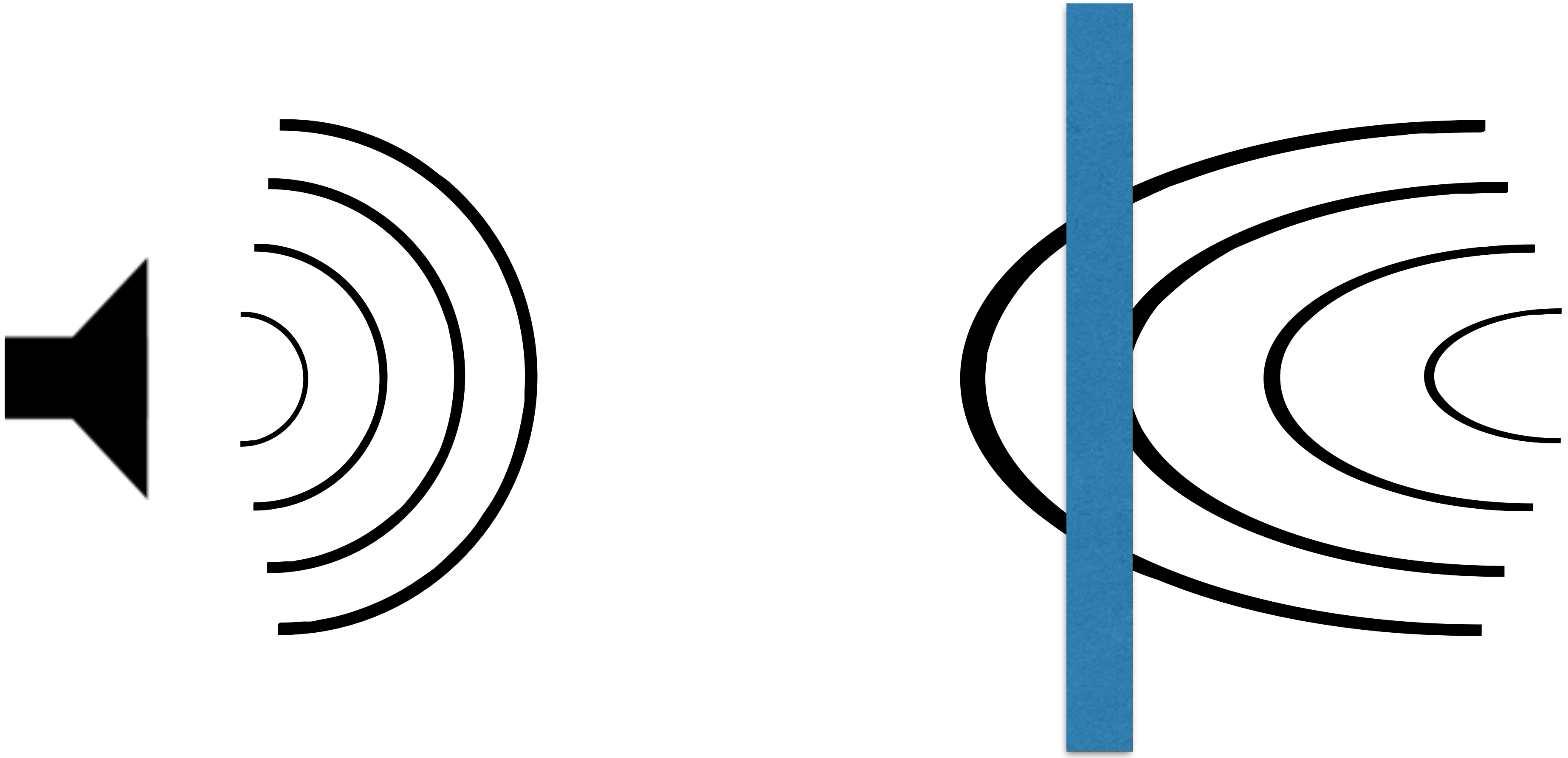
the doppler effect



the doppler effect



the doppler effect



the doppler effect

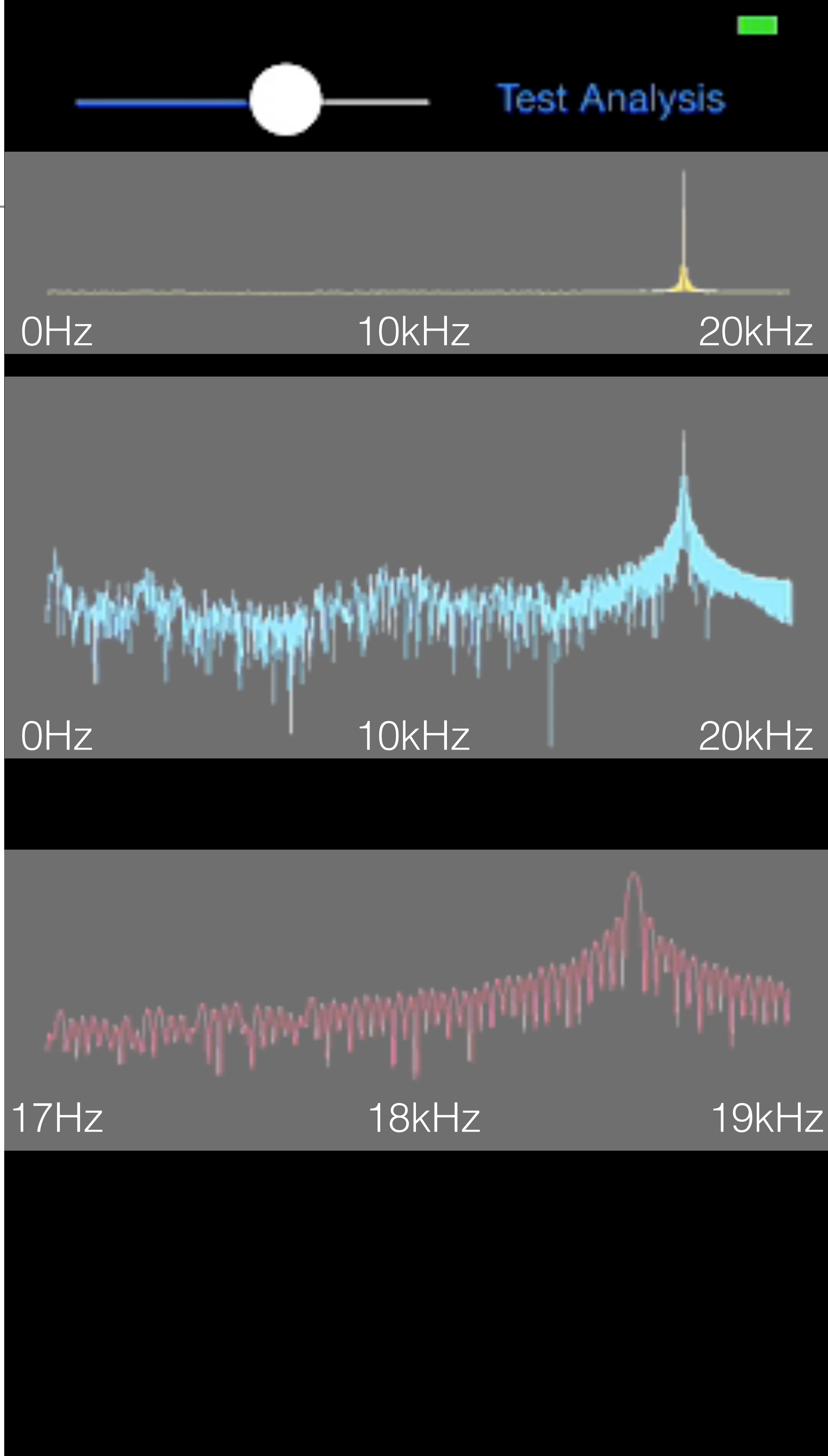
change in frequency

velocity of object

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

speed of sound

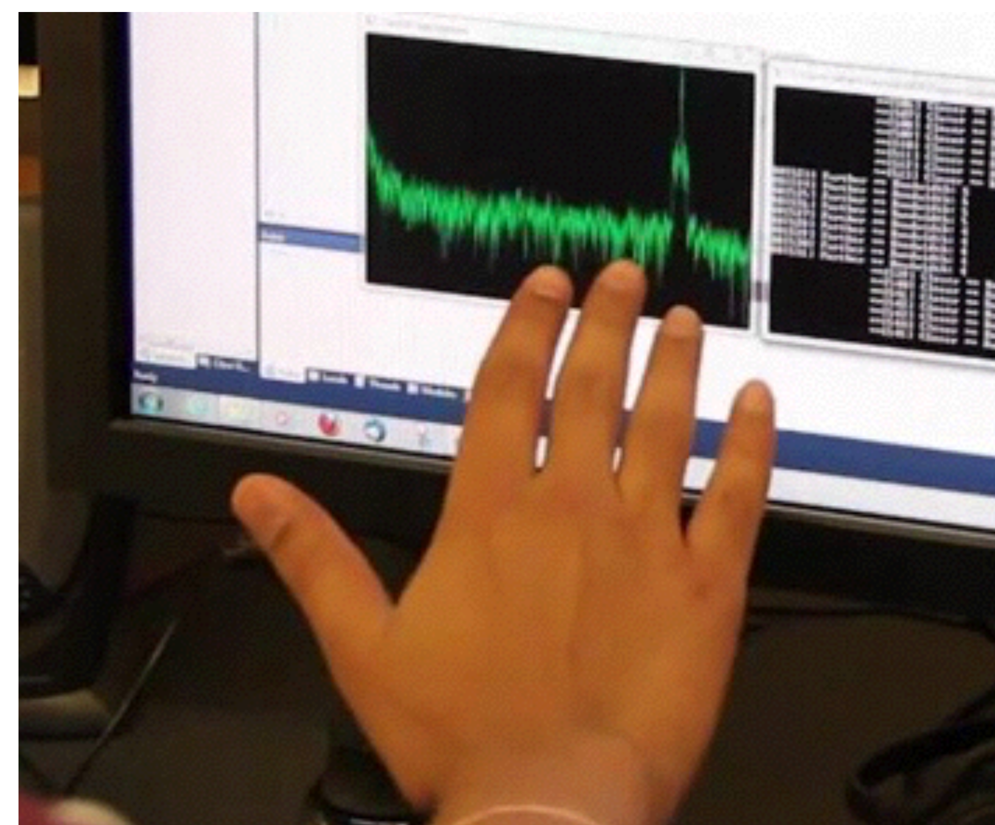
frequency of source



linear

db

db zoomed (freq axis)



A2 specifications

Module A: Create an iOS application using the example template that:

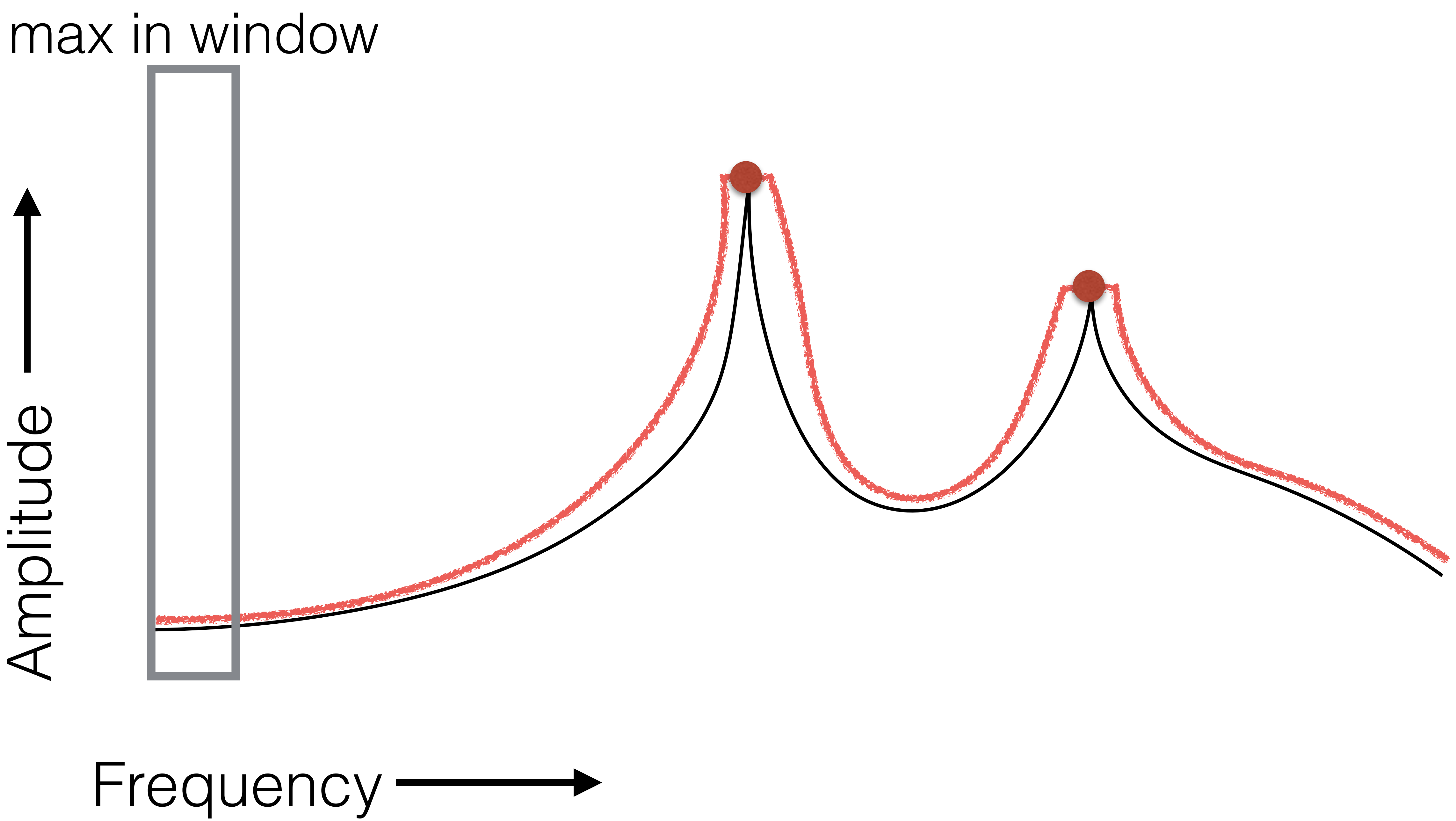
- Reads from the microphone
- Takes an FFT of the incoming audio stream
- Displays the frequency of the two loudest tones within 6Hz accuracy (+-3Hz)
  - Please have a way to "lock in" the last frequencies detected on the display
- Is able to distinguish tones at least 50Hz apart, lasting for 200ms or more
- **Exceptional**: recognize two tones played on a piano (down to one half step apart) and report them by letter (*i.e.*, A4, A#4). Must work at note A2 and above. **Note**: this is harder than just identifying two perfect sine waves!!

The sound source must be external (*i.e.*, laptop, instrument, another phone, *etc*.).

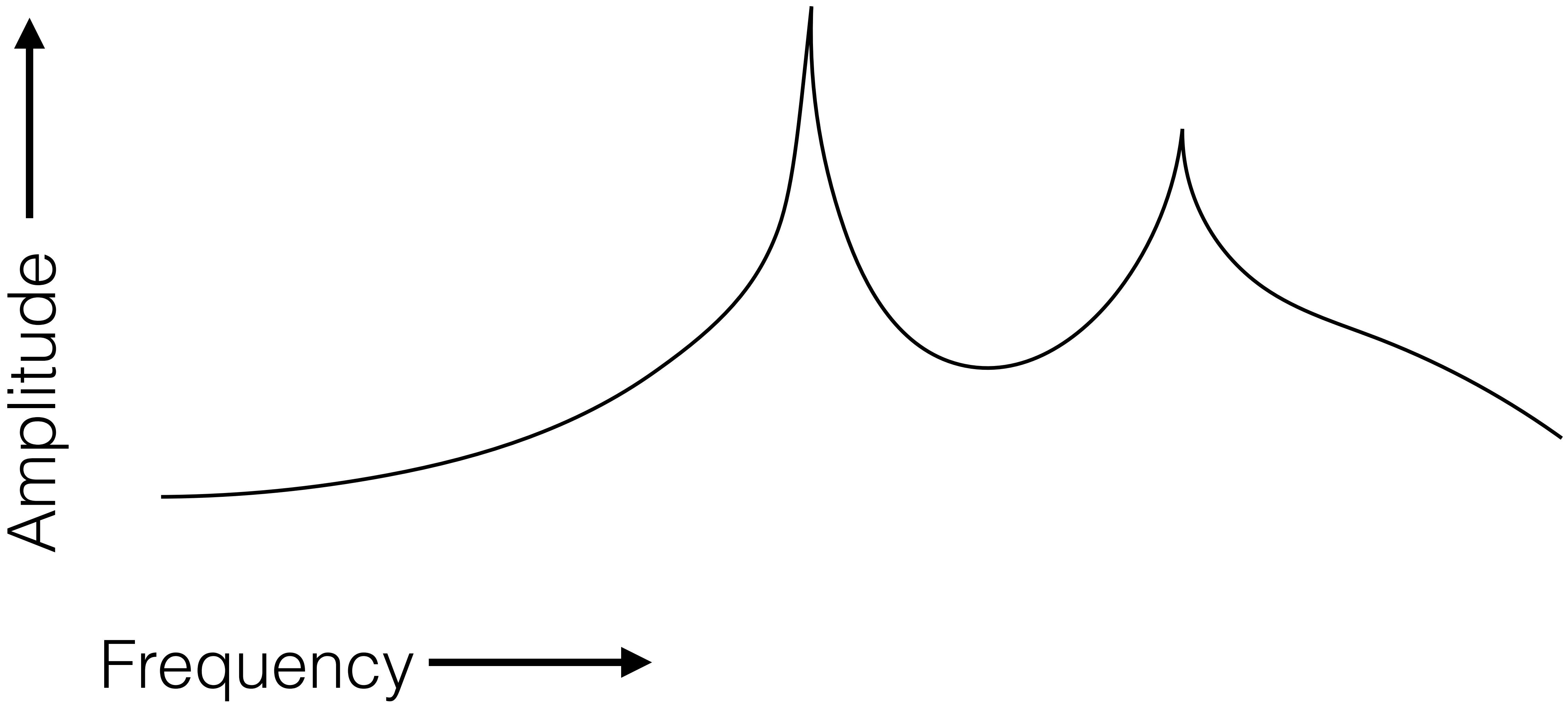
Module B: Create an iOS application using the example template that:

- Reads from the microphone
- Plays a settable (via a slider or setter control) inaudible tone to the speakers (15-20kHz)
- Displays the magnitude of the FFT of the microphone data in decibels
- Is able to distinguish when the user is {**not gesturing**, **gestures toward**, or **gesturing away**} from the microphone using Doppler shifts in the frequency

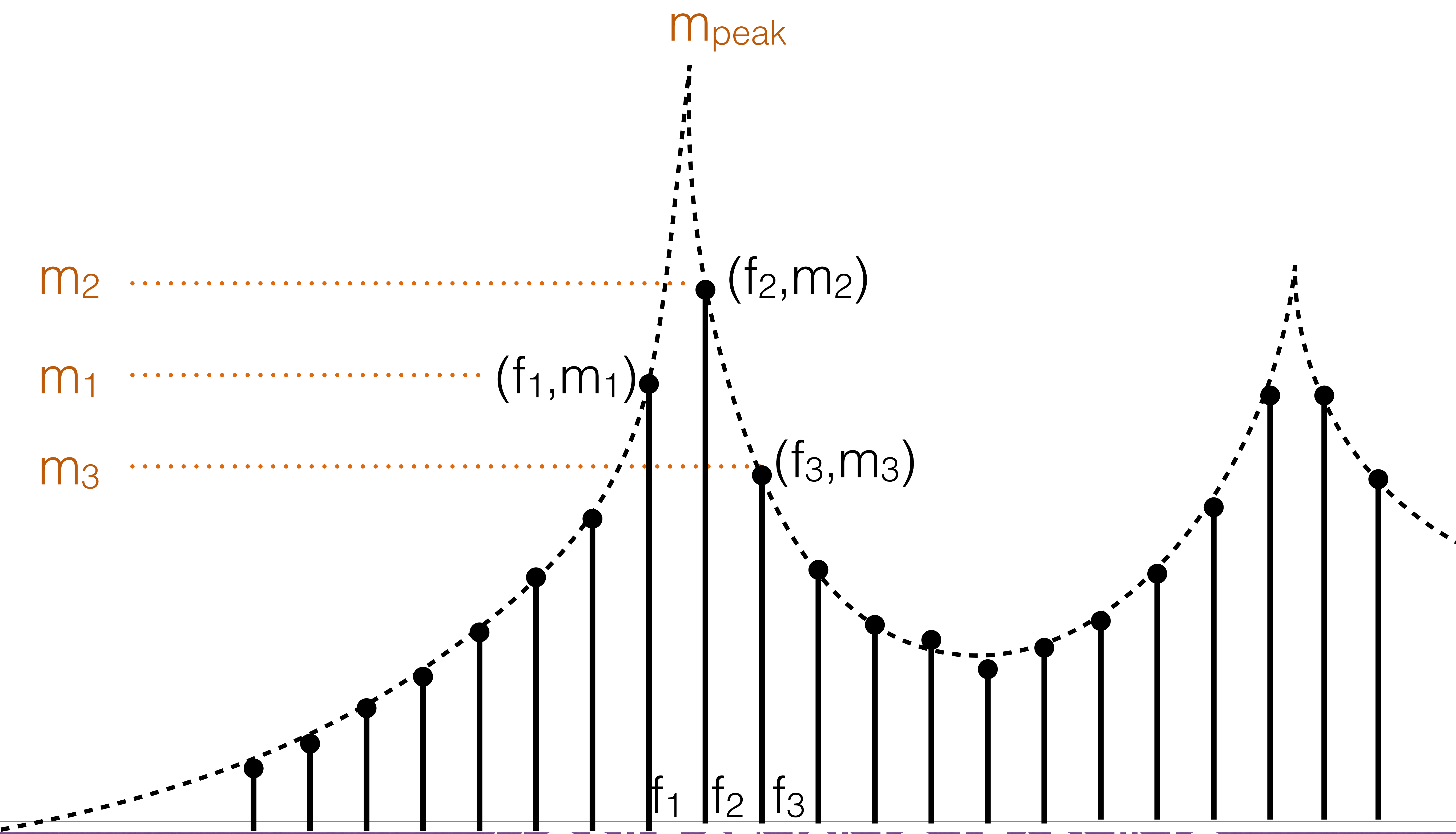
local peak finding



peak interpolation



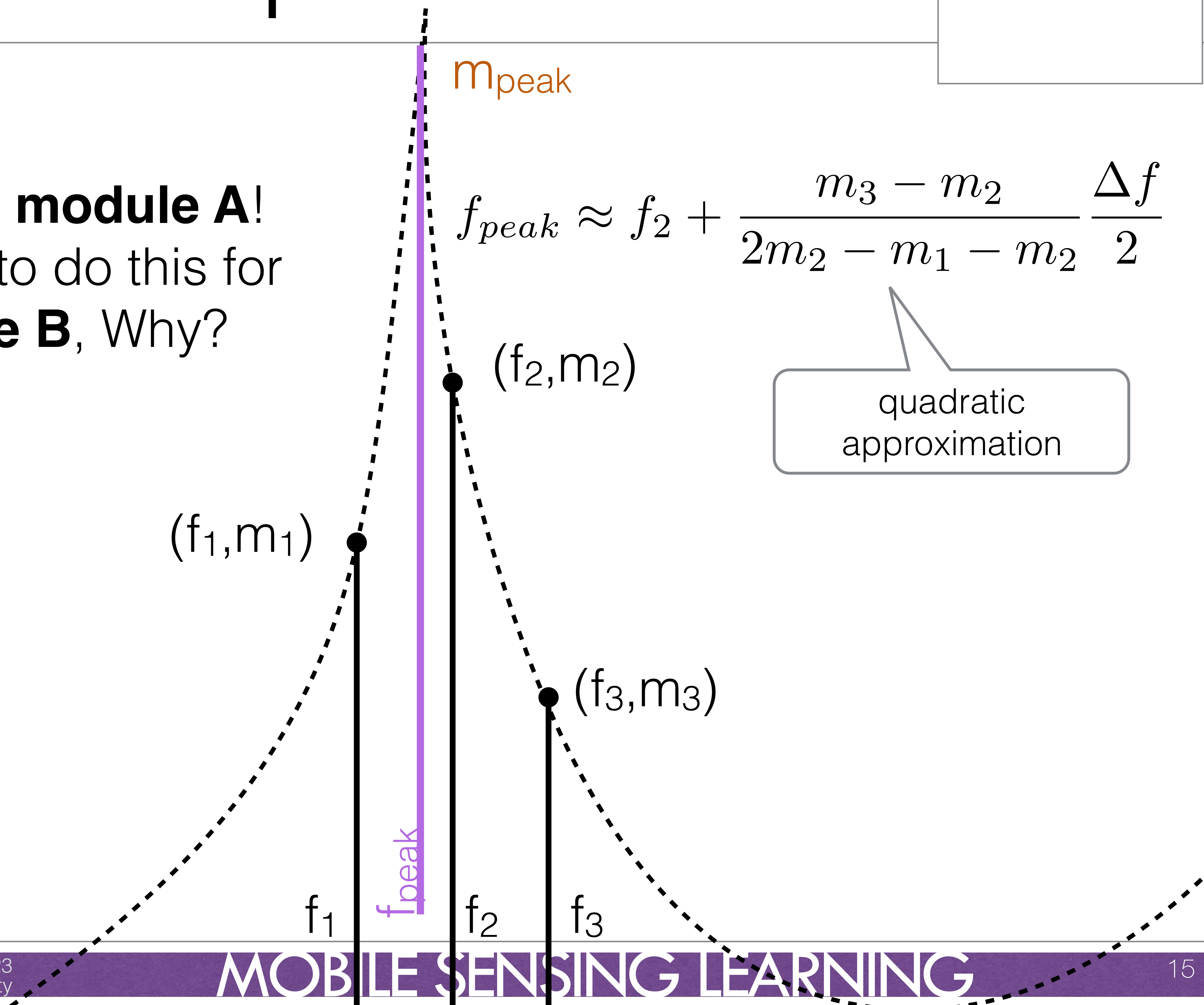
peak interpolation



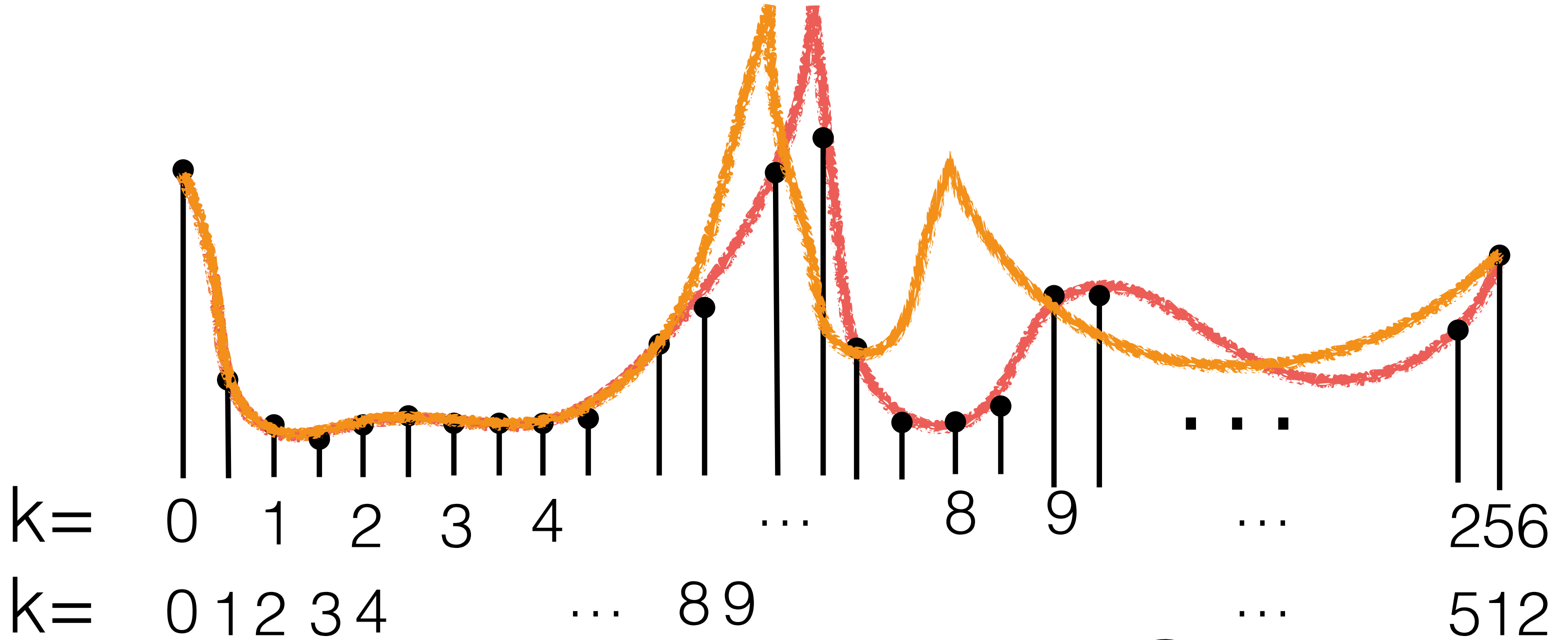
peak interpolation

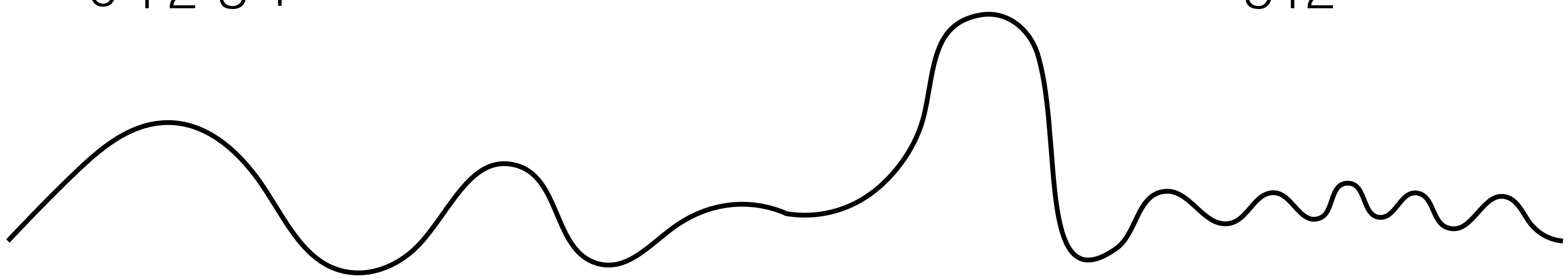
Great for **module A**!
No need to do this for **module B**, Why?

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



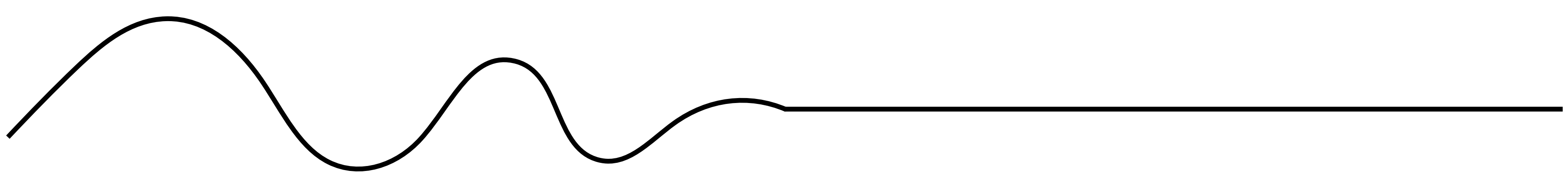
resolution for the FFT





256 points

next 256 points=512 total points

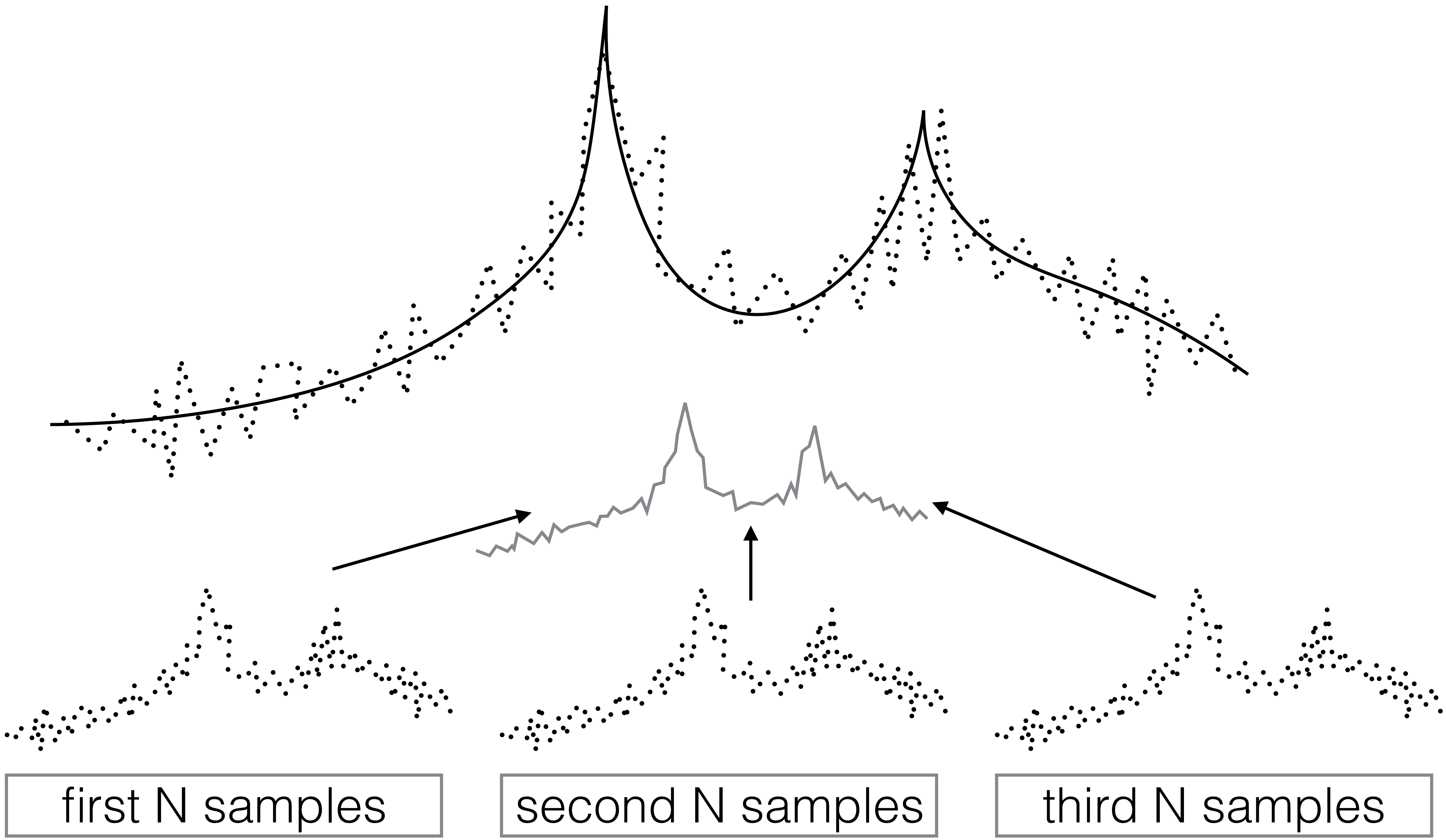


solution!

zero padding

noise in the FFT

- variance around actual magnitude unavoidable



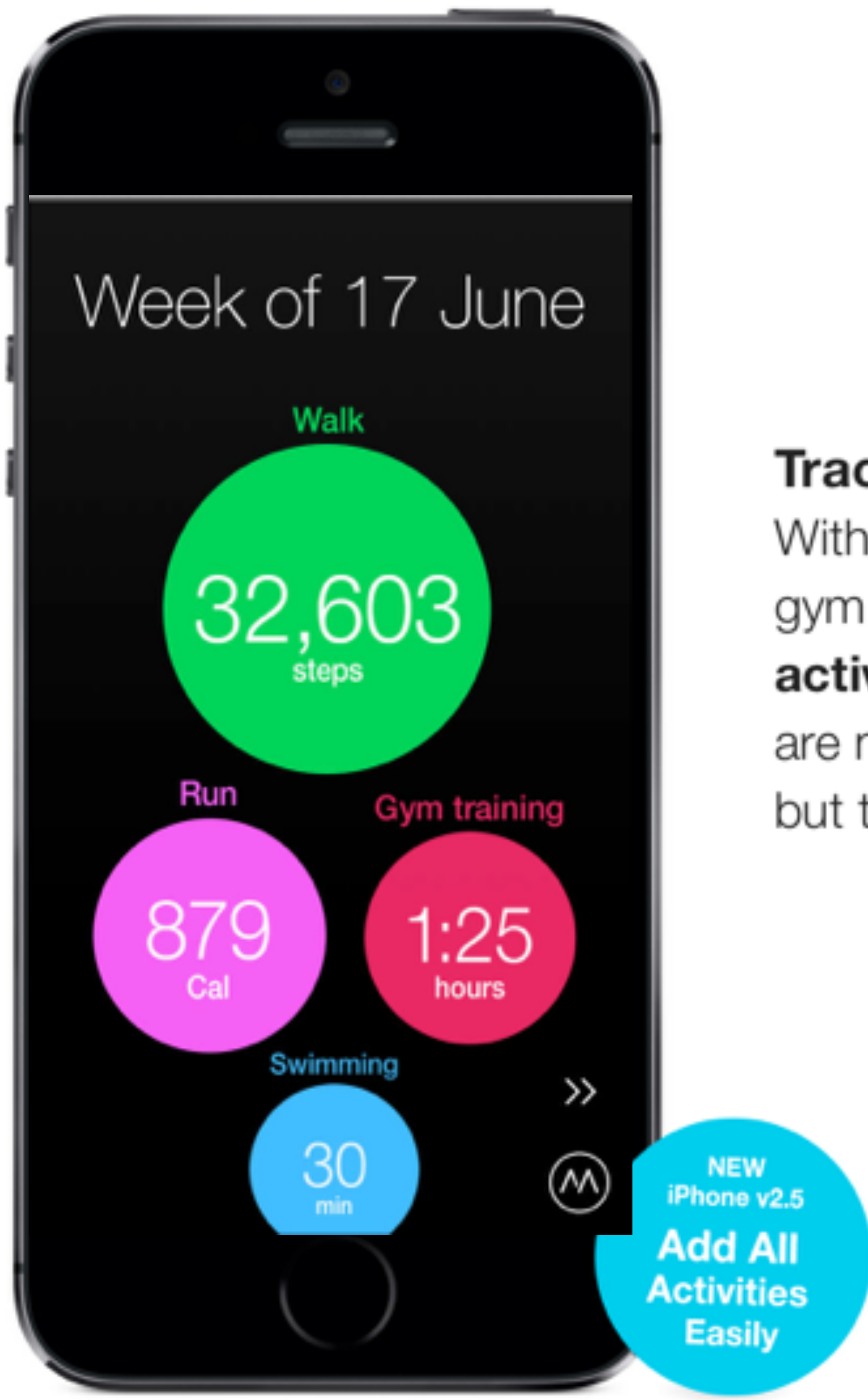
Questions on the FFT

- 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

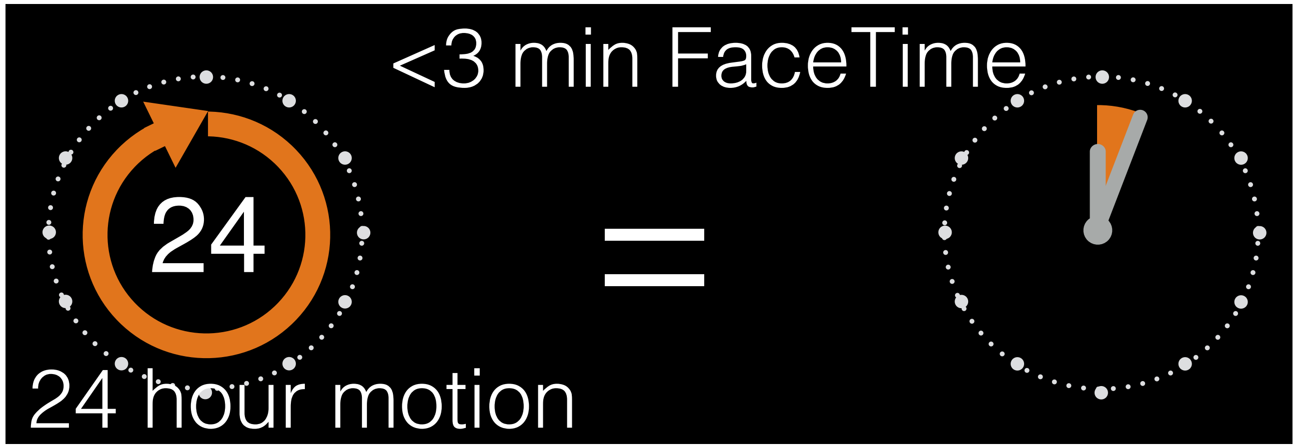


Track all activity*
With Moves 2.5 for iPhone, you can add gym training and **over 60 other activities** by duration. These activities are not (yet!) automatically recognized, but they are easy to add.

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)

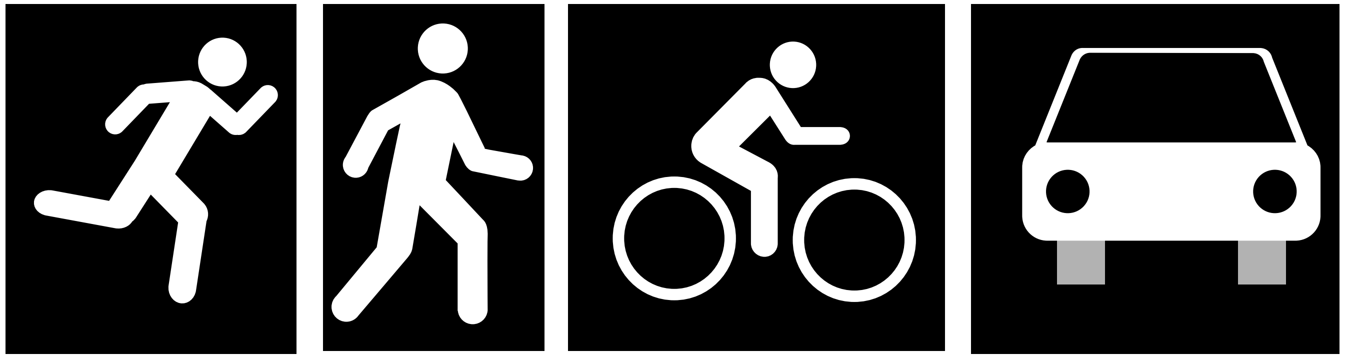
- mediates all access to data
    - battery life++
    - parallel processing++
    - overhead += 0, seriously



- sensor fusion for more accurate analysis, very cool

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];
    }


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

    }
    else
        NSLog(@"Cannot start activity manager");


if([CMMotionActivityManager isActivityAvailable] == YES )
        [self.motionActivityManager stopActivityUpdates];
```

import framework

declare activity manager

device capable?

instantiate

subscribe

queue to run on

block to handle updates

end subscription

swift version

```swift
import CoreMotion

let activityManager = CMMotionActivityManager()
let customQueue = NSOperationQueue() // not the

 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

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

what's in an update?

Motion Activity

Running

Completely insensitive to location

Shortest latency

Most accurate classification

what's in an update?

Motion Activity

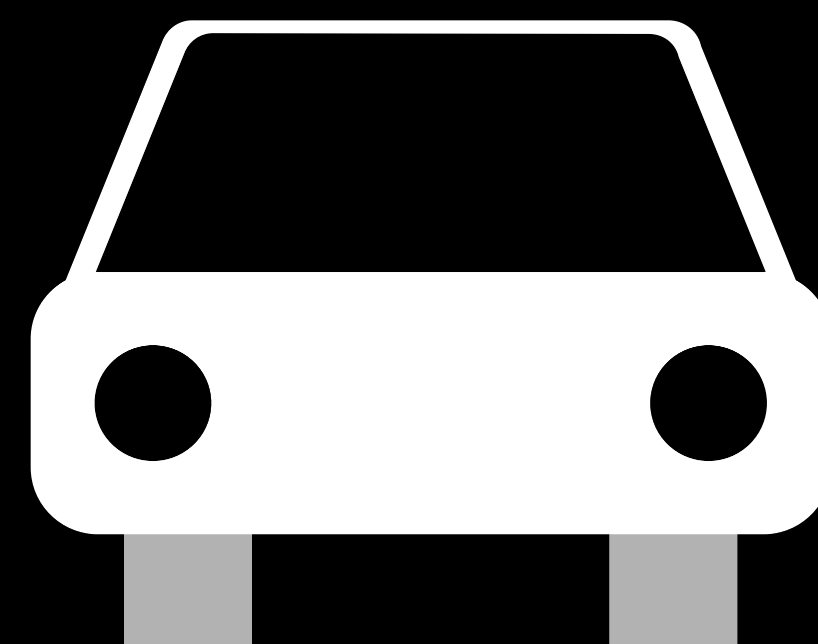
Automotive

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



what's in an update?

Motion Activity

Cycling

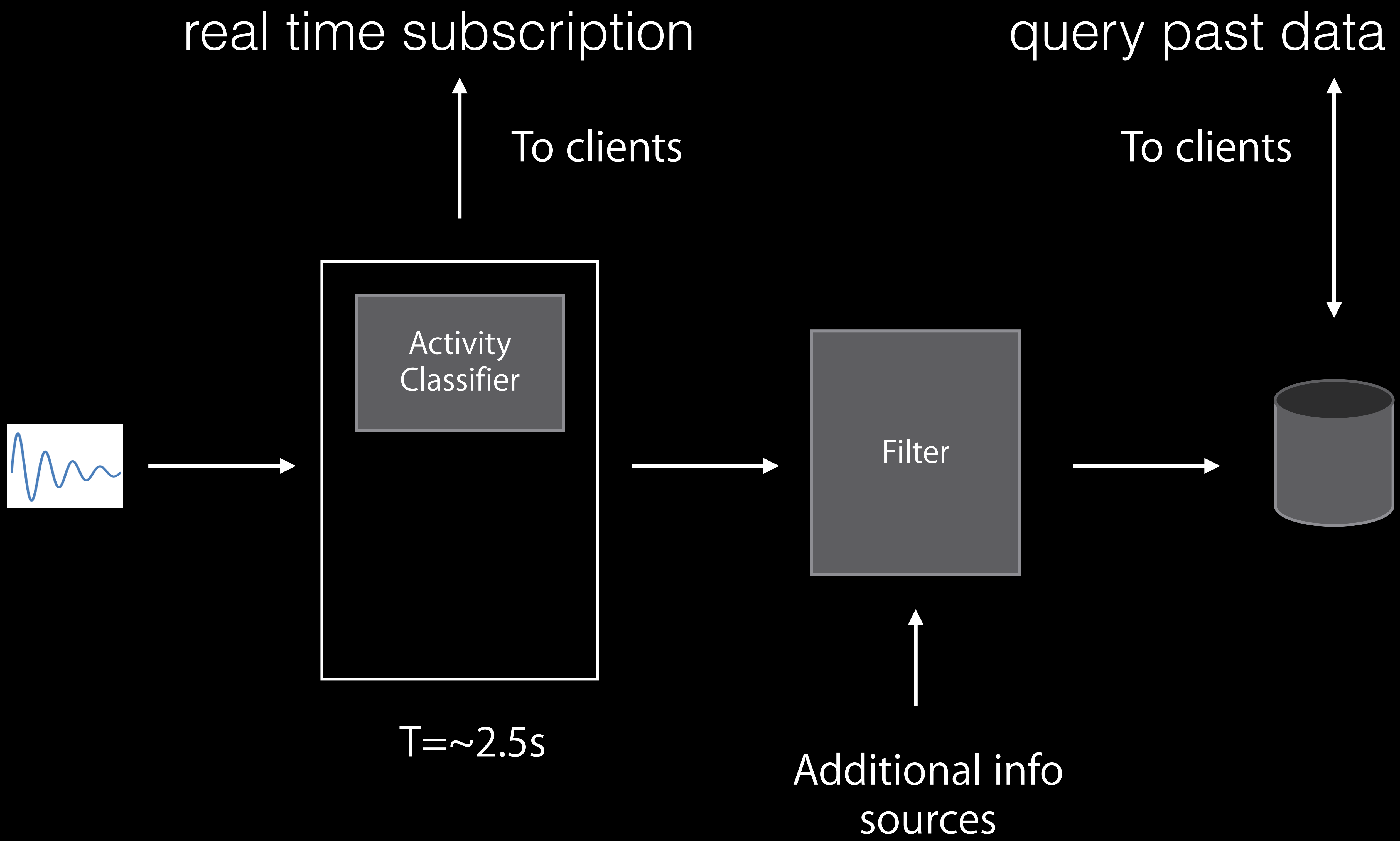
Performance is very sensitive to location

- Works best if device is worn on upper arm

Longest latency

- Best for retrospective use cases

Motion Processing Architecture



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

swift

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)
        }
}
```

closure handler for updates

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

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

```
let pedometer = CMPedometer()

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


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

declare and init

available on this device?

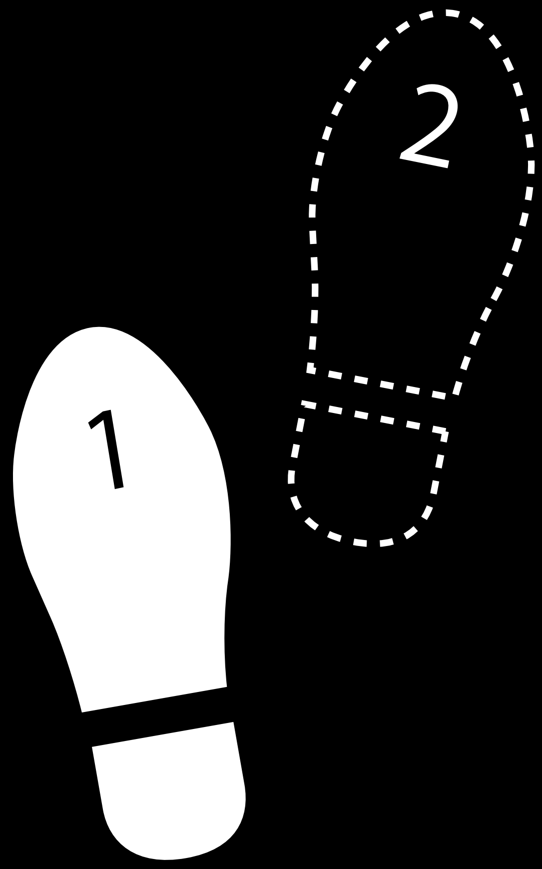
properties from step counter

unsubscribe

CMPedometerData,<startDate 2016-09-06 17:13:54 +0000, endDate 2016-09-06 17:14:21 +0000, steps 16, distance 14.1199999999534, floorsAscended 0, floorsDescended 0 currentPace 0.328659236431218 currentCadence 3.29512763023765>

Pedometer

Step counting



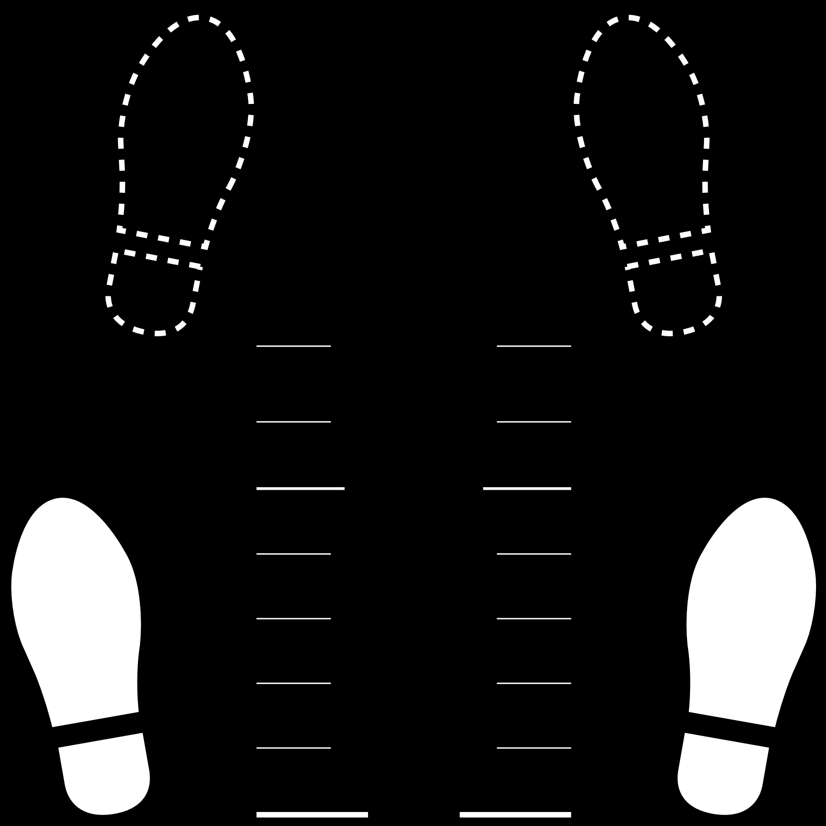
Consistent performance across body locations

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

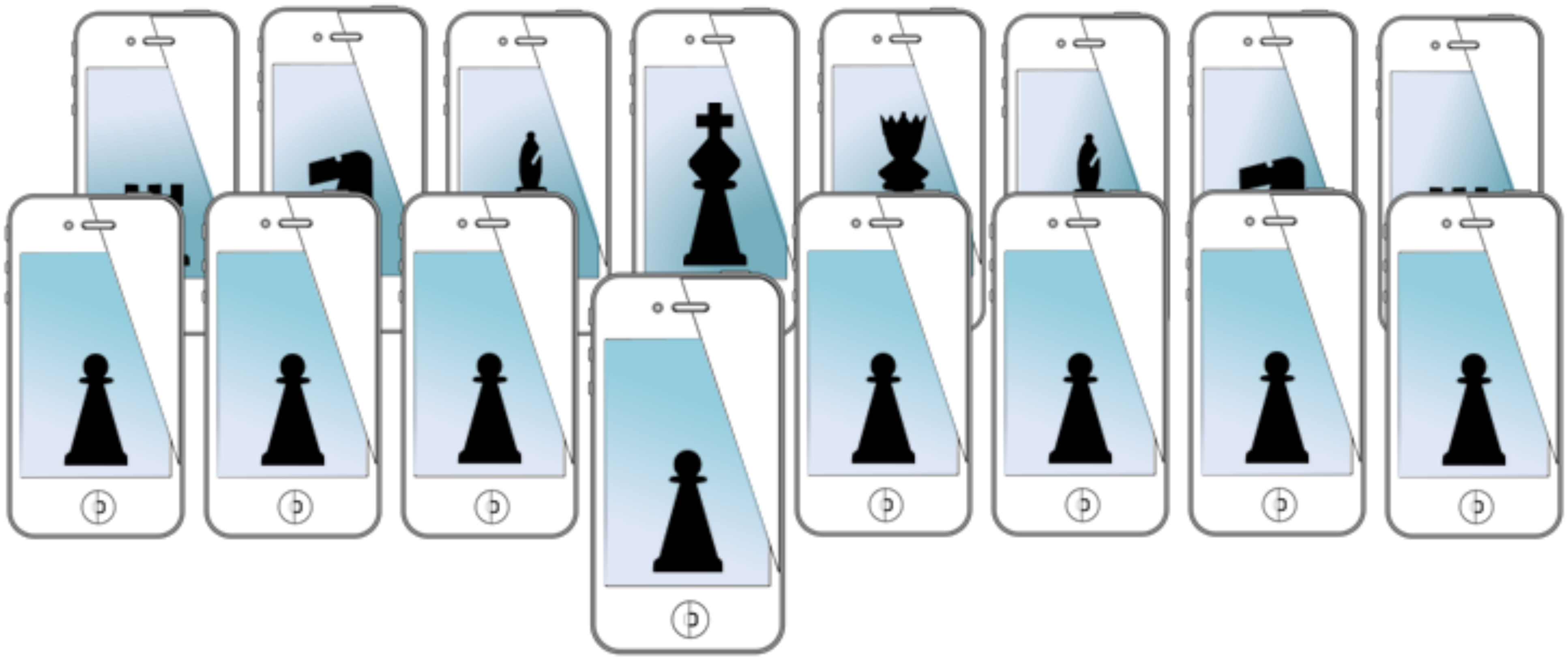
querying past steps

```
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
    }
}
```

handle error!

access properties

MOBILE SENSING LEARNING



CSE5323 & 7323

Mobile Sensing and Learning

week 5, lecture a: doppler and activity monitoring

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