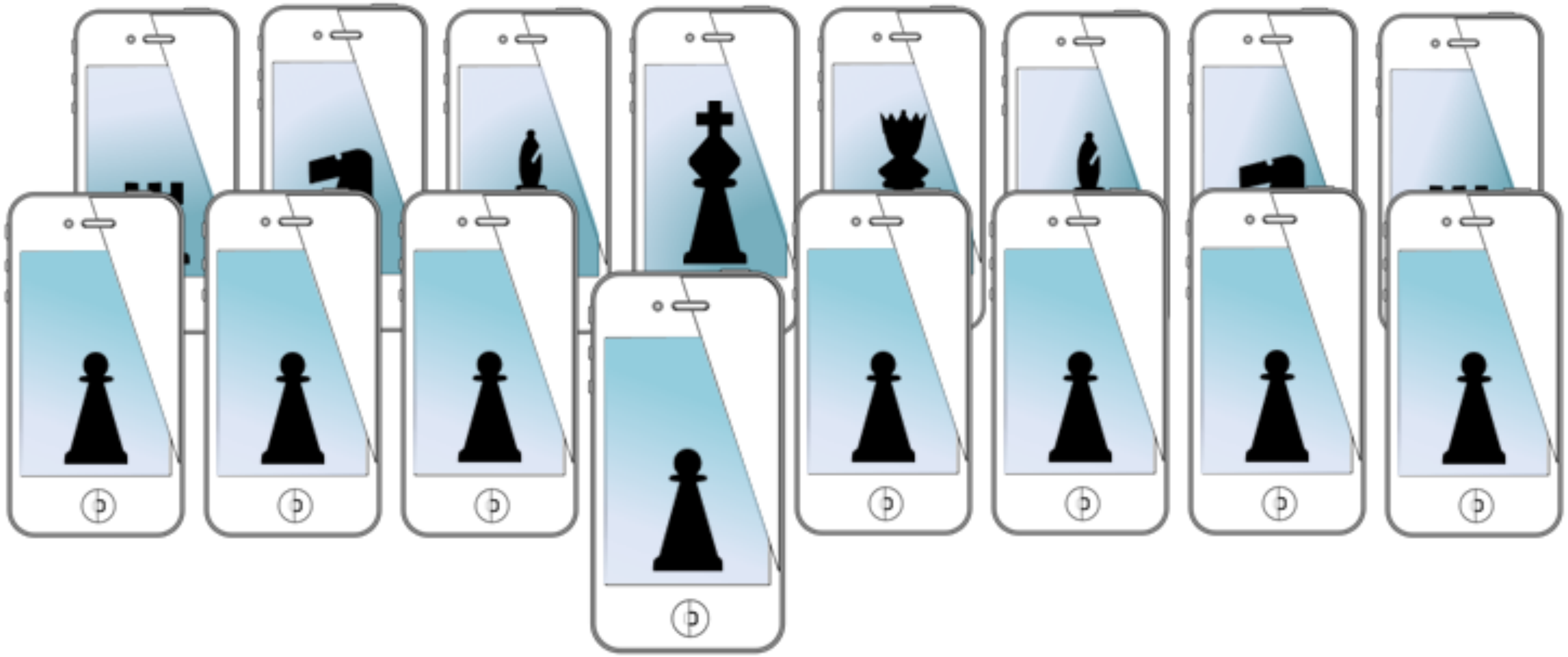
MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

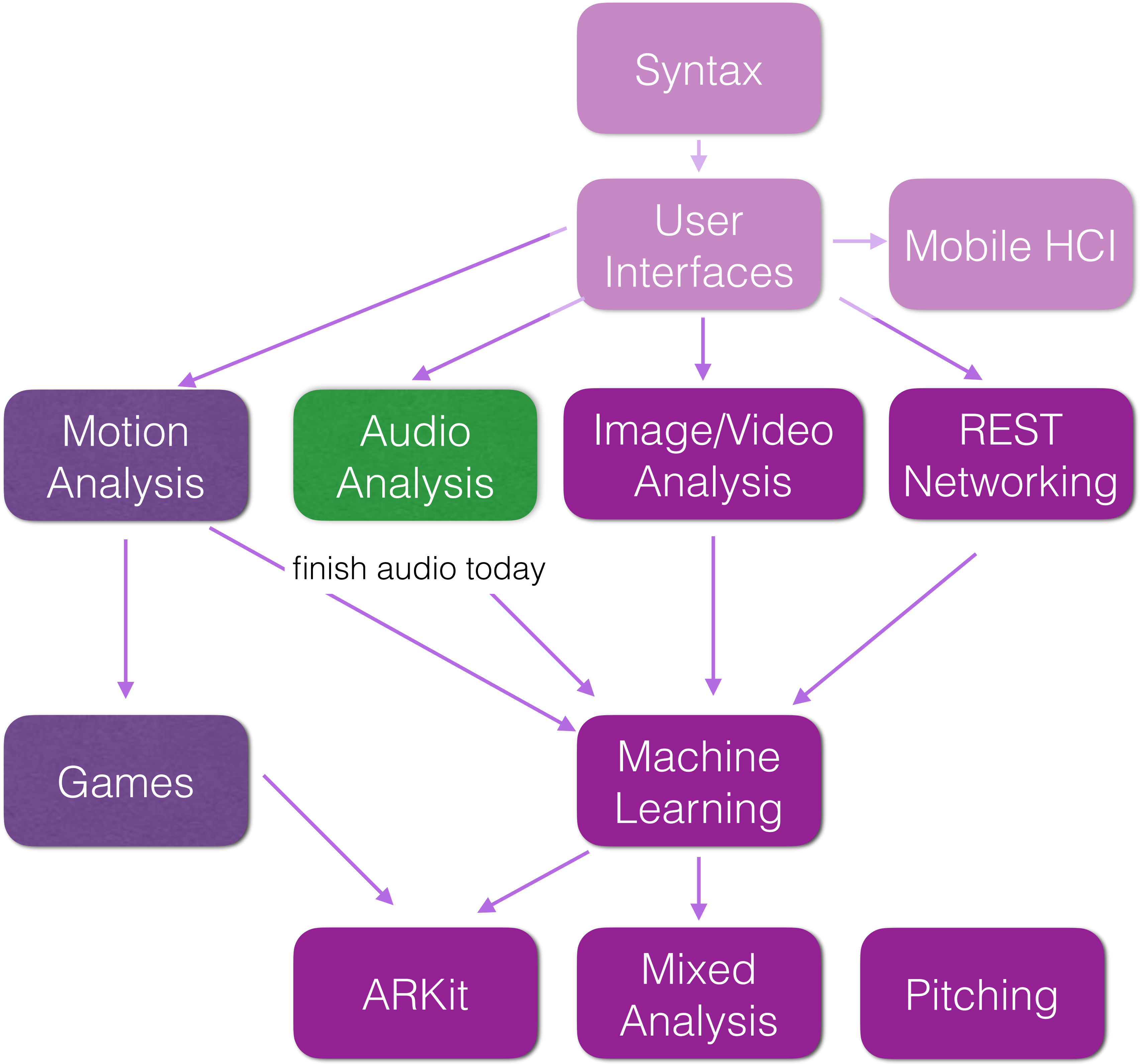
doppler and activity monitoring

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

agenda and logistics

- logistics:
  - grades update
  - A2 is due soon!
- agenda:
  - A2 explanations
    - general FFT review
    - peak finding
    - the doppler effect
  - activity processing

class overview



finish audio today

A2 specifications

On Canvas

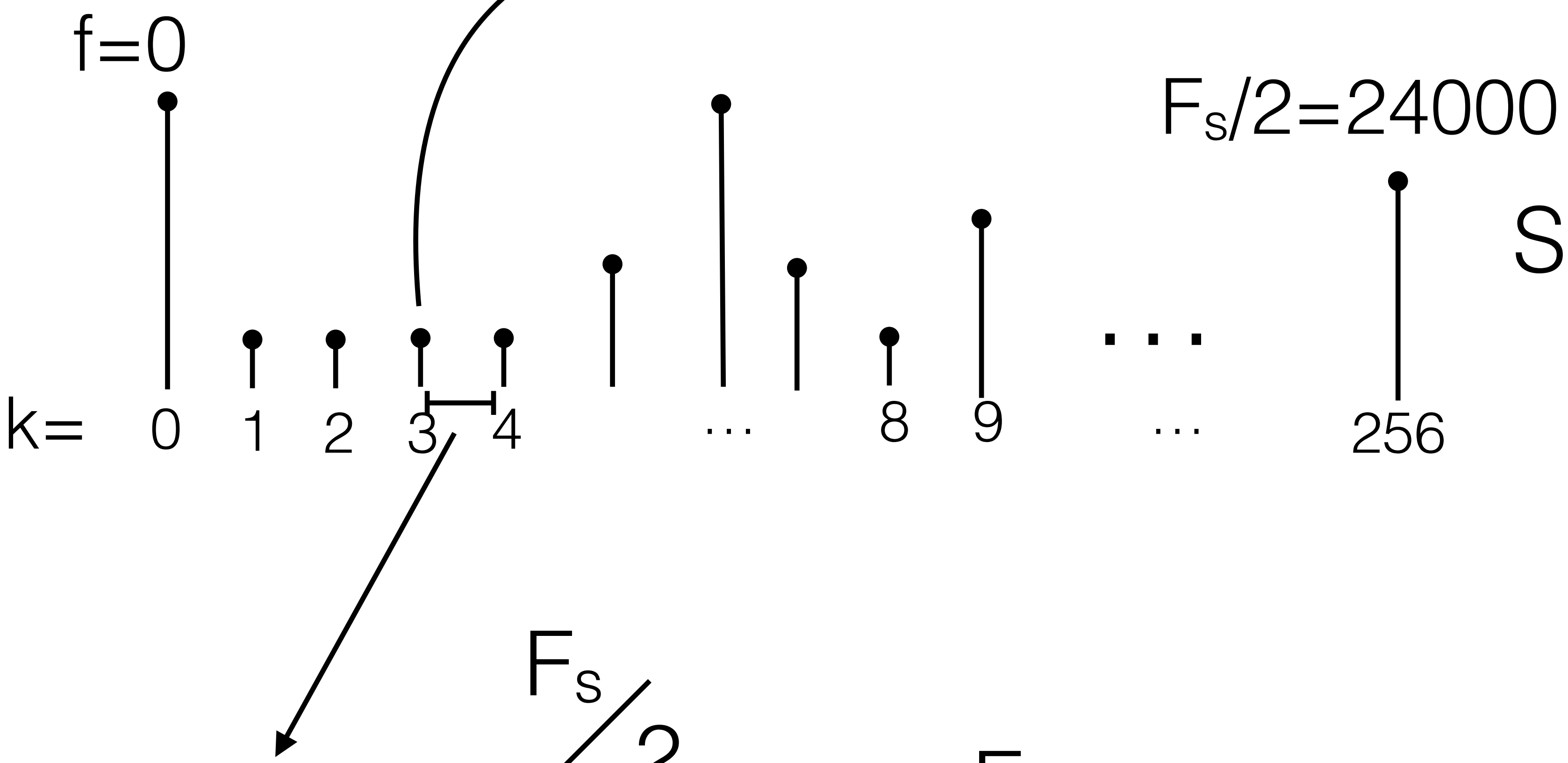
FFT review

- sampling rate
  - dictates the time between each sample, (1 / Fs)
  - 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

Note: the FFT class **ALWAYS** rounds to the next power of 2

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



time array, N = 512

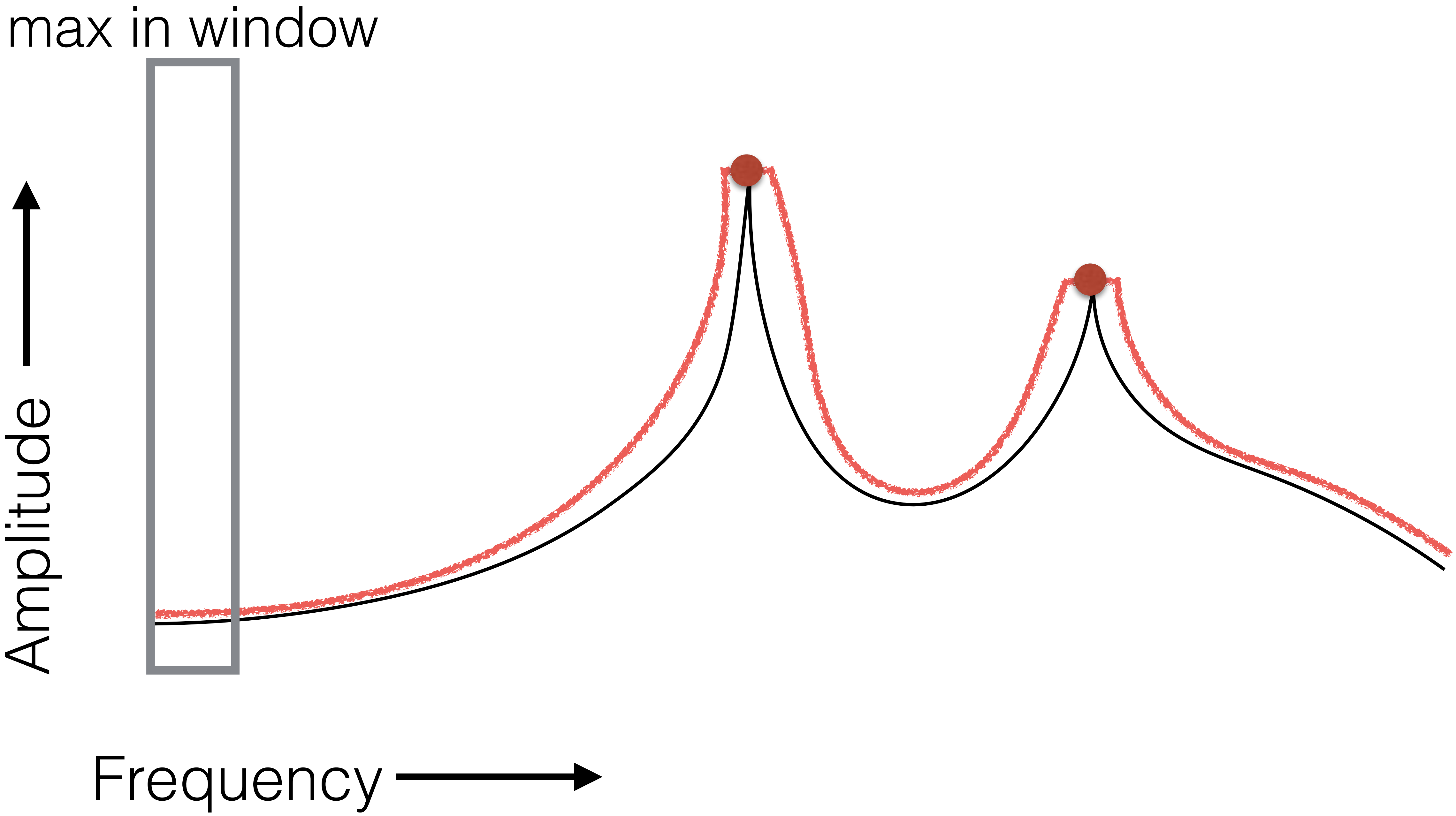
FFT array, N = 512

Spectrum, N/2 = 256

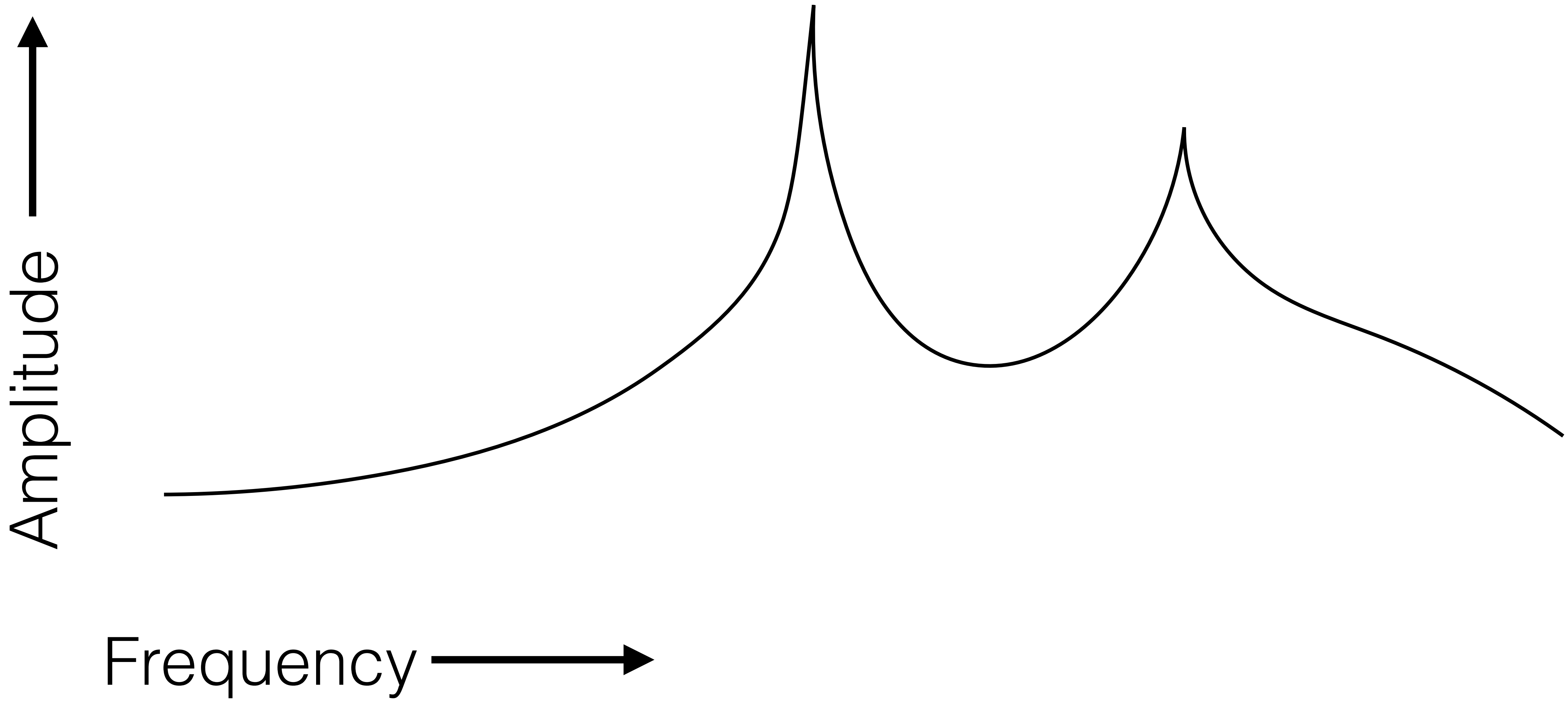
Fs = 48000 Hz

$$df = \frac{{}^{F_s}\!/_{2}}{{}^{N}\!/_{2}} = {}^{F_s}\!/_{N} = 48000/512 = \sim 93.75 \text{ Hz}$$

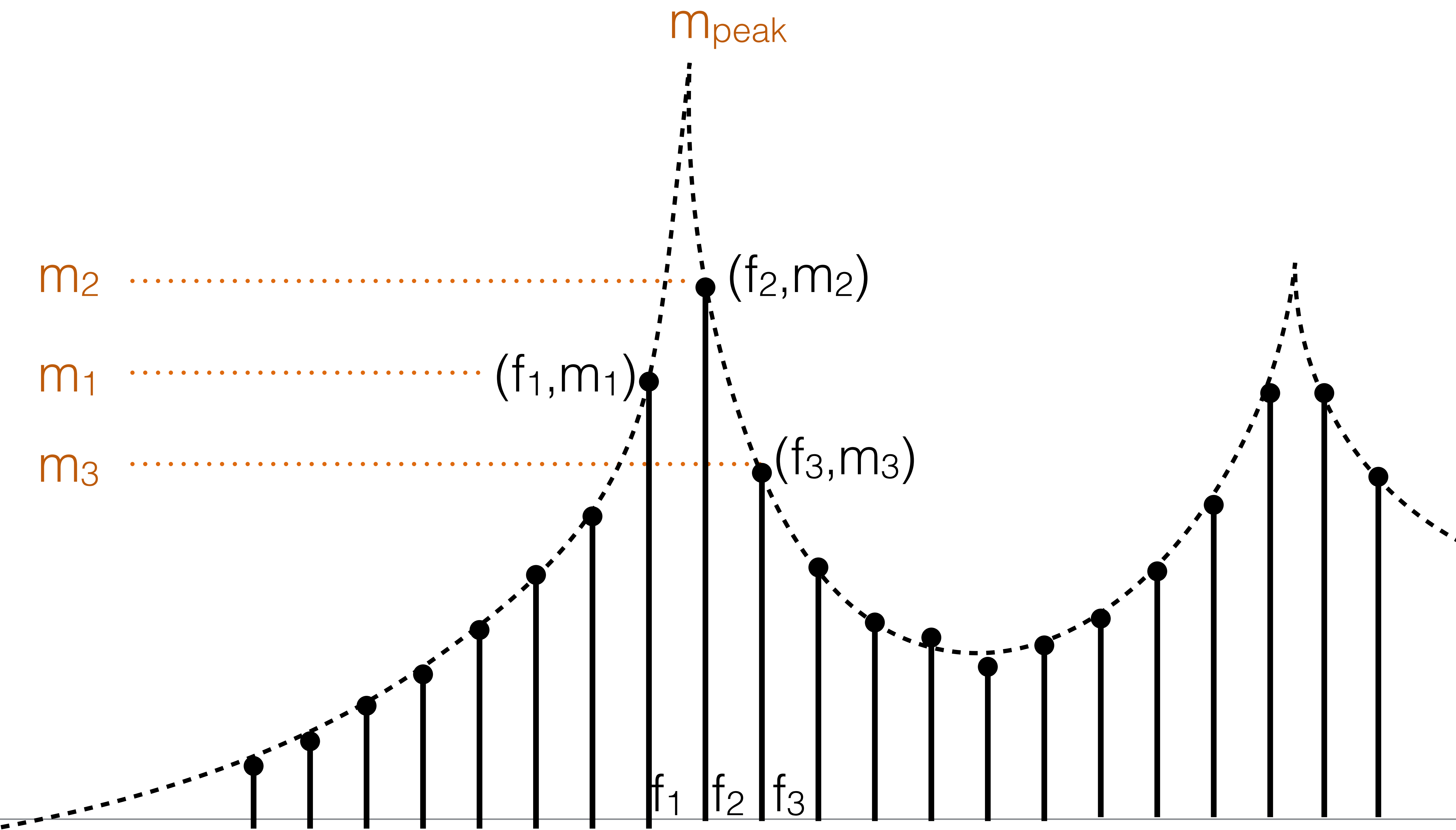
local peak finding



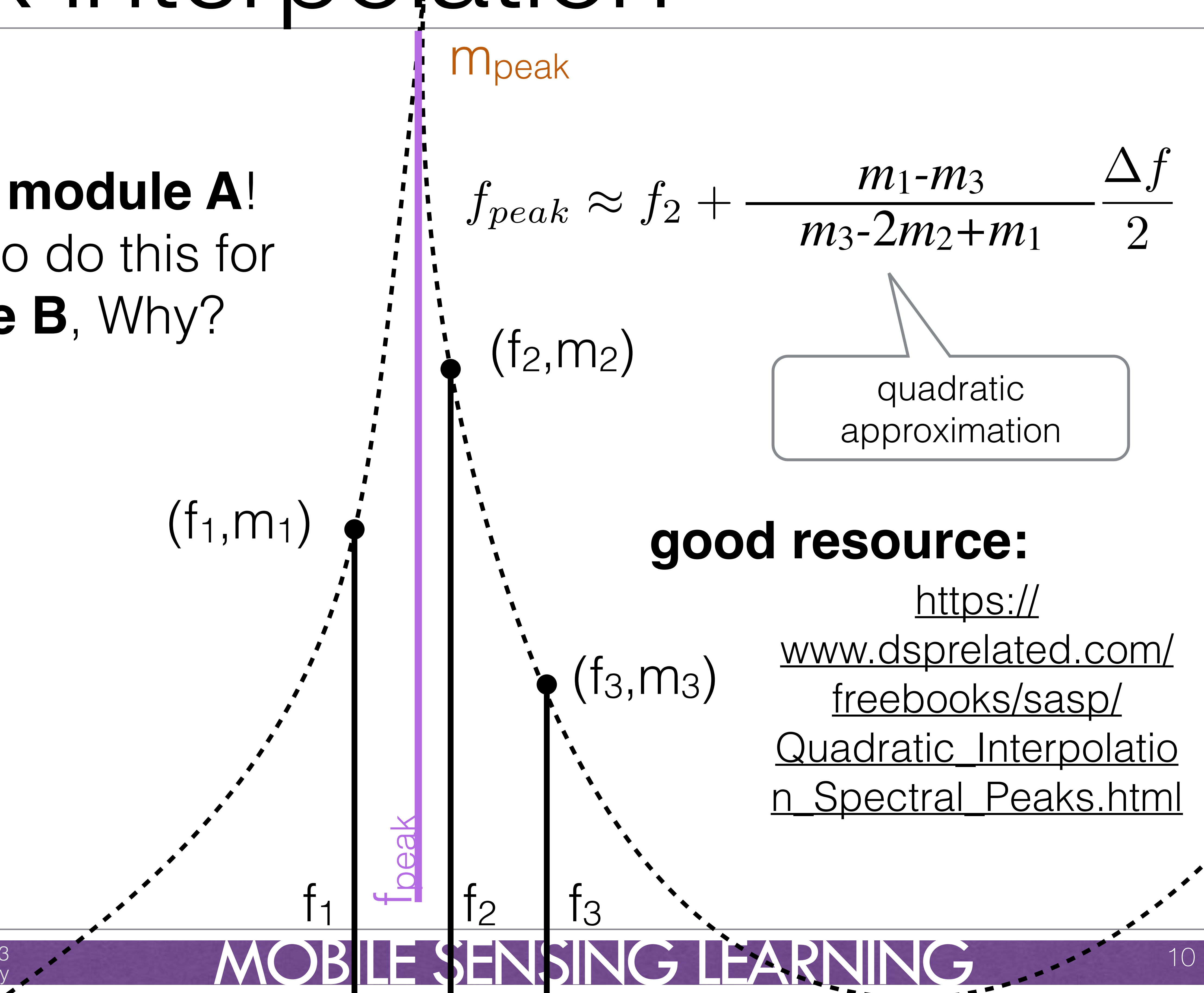
peak interpolation



peak interpolation



peak interpolation



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

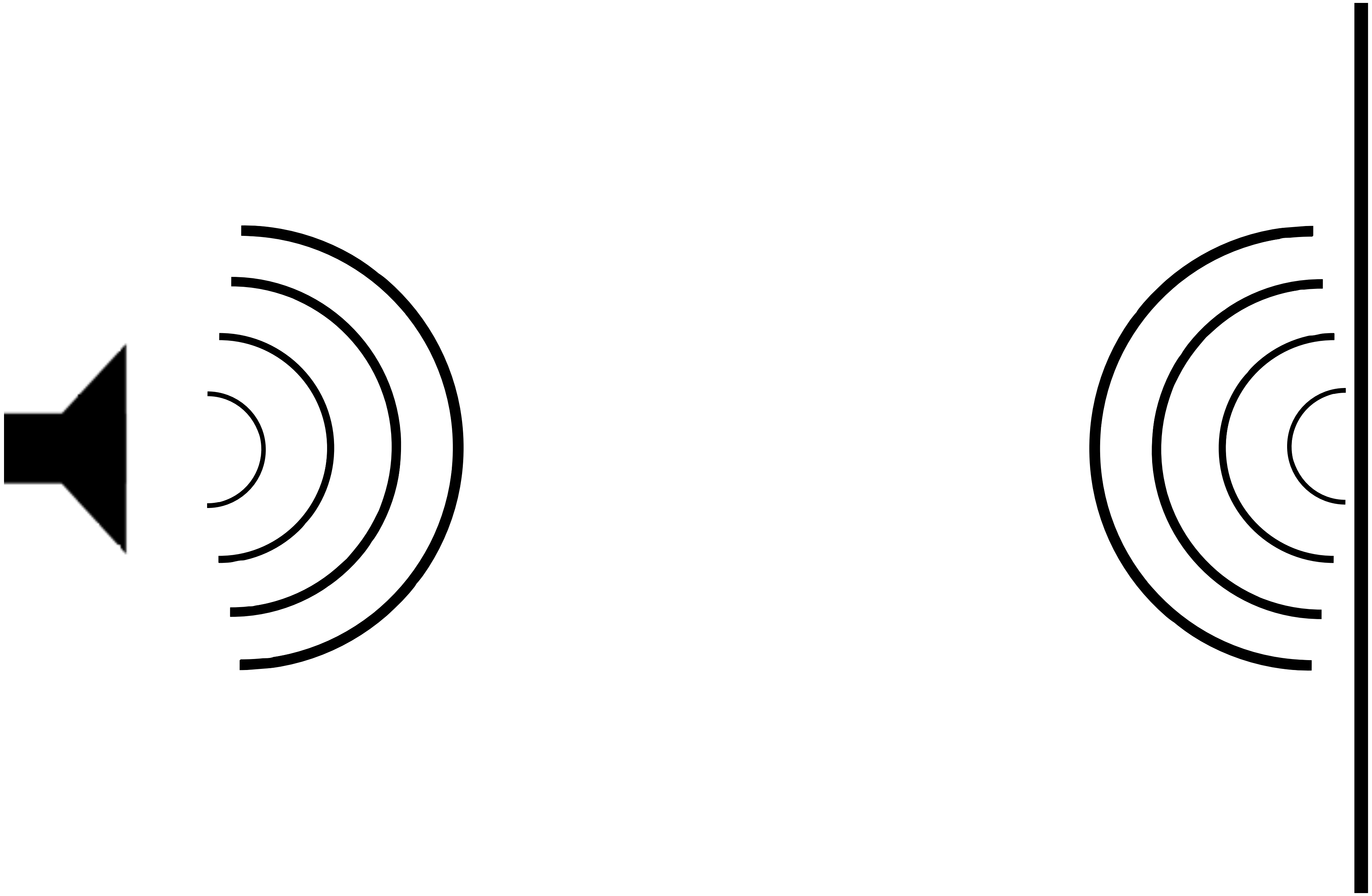
$$f_{peak} \approx f_2 + \frac{m_1 - m_3}{m_3 - 2m_2 + m_1} \frac{\Delta f}{2}$$

quadratic approximation

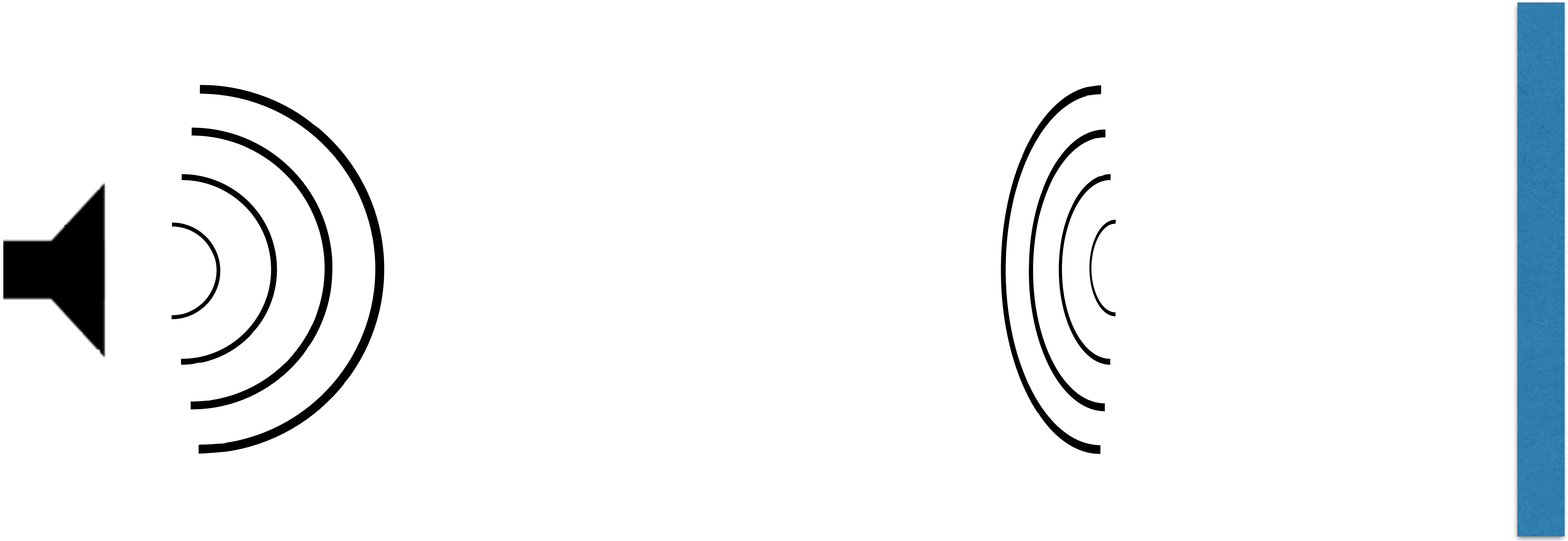
good resource:

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

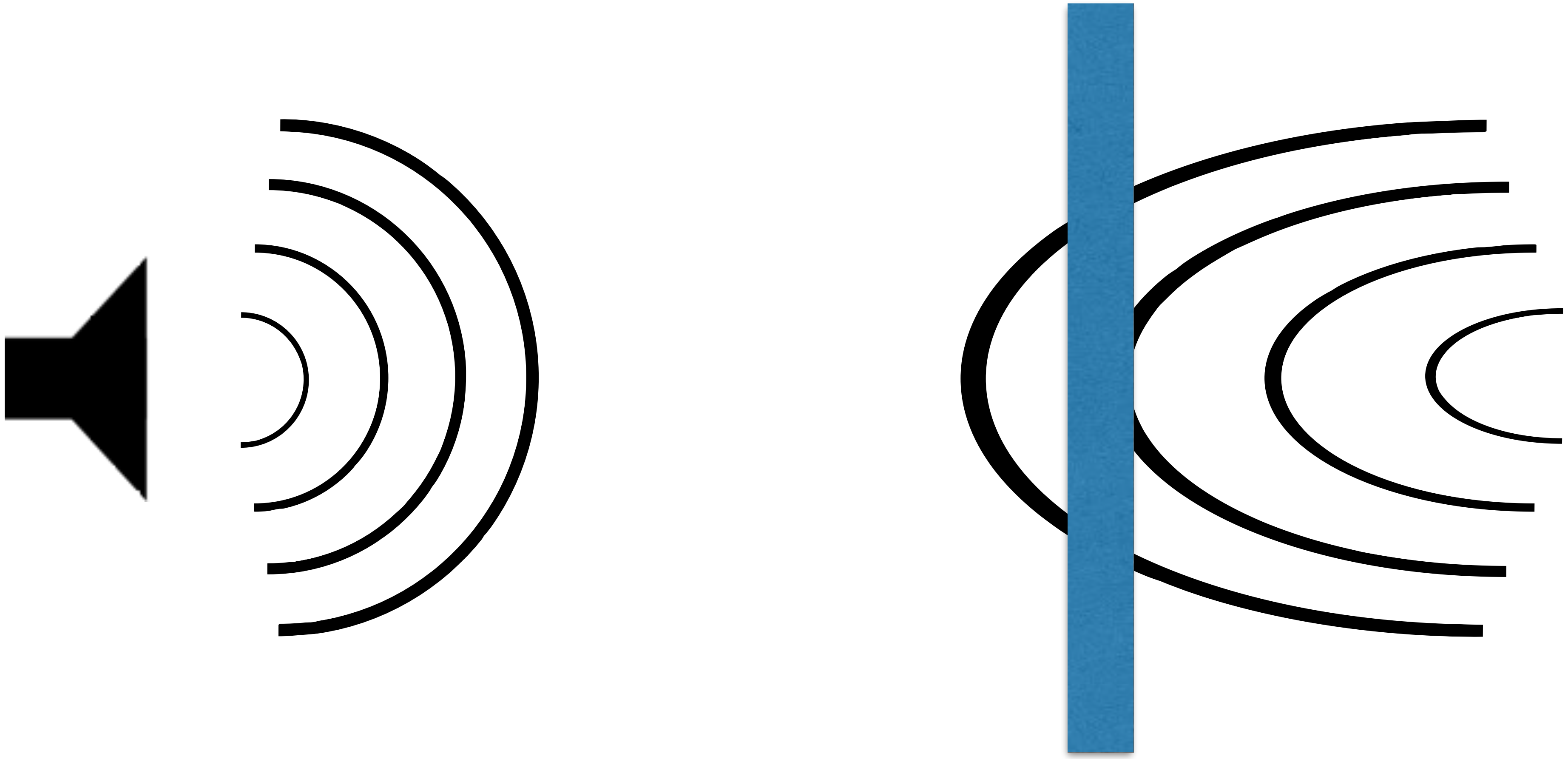
the doppler effect



the doppler effect



the doppler effect



the doppler effect

velocity of object

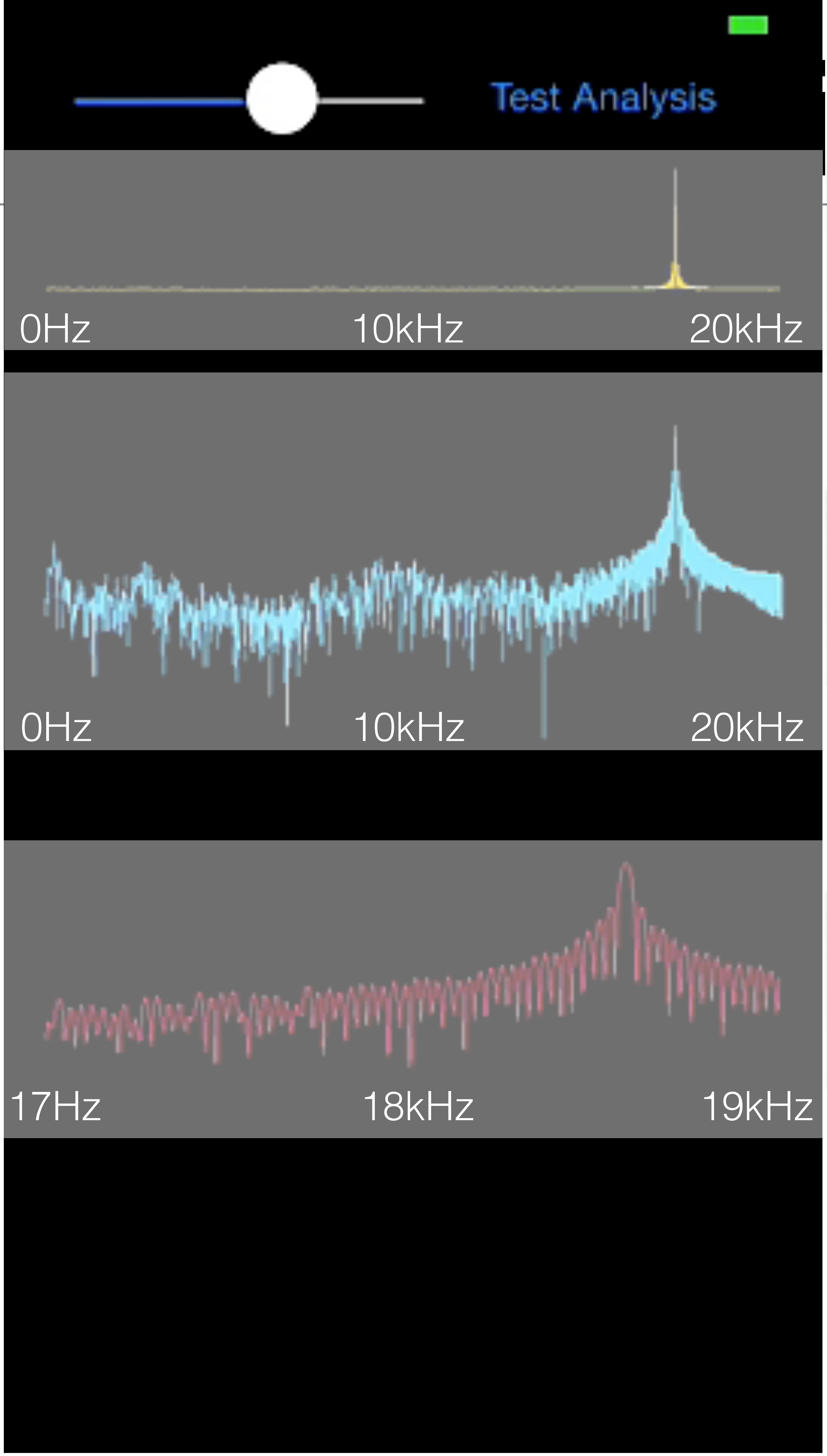
change in frequency

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

speed of sound

frequency of source

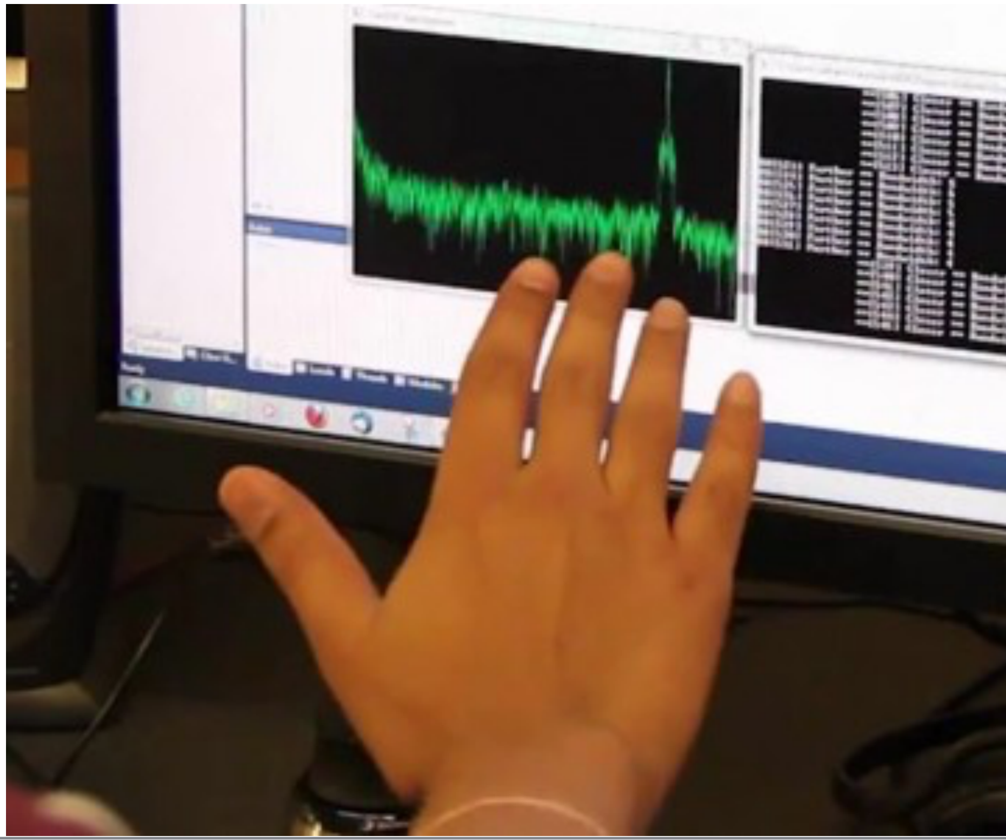
ifts fro



linear

db

db zoomed (freq axis)

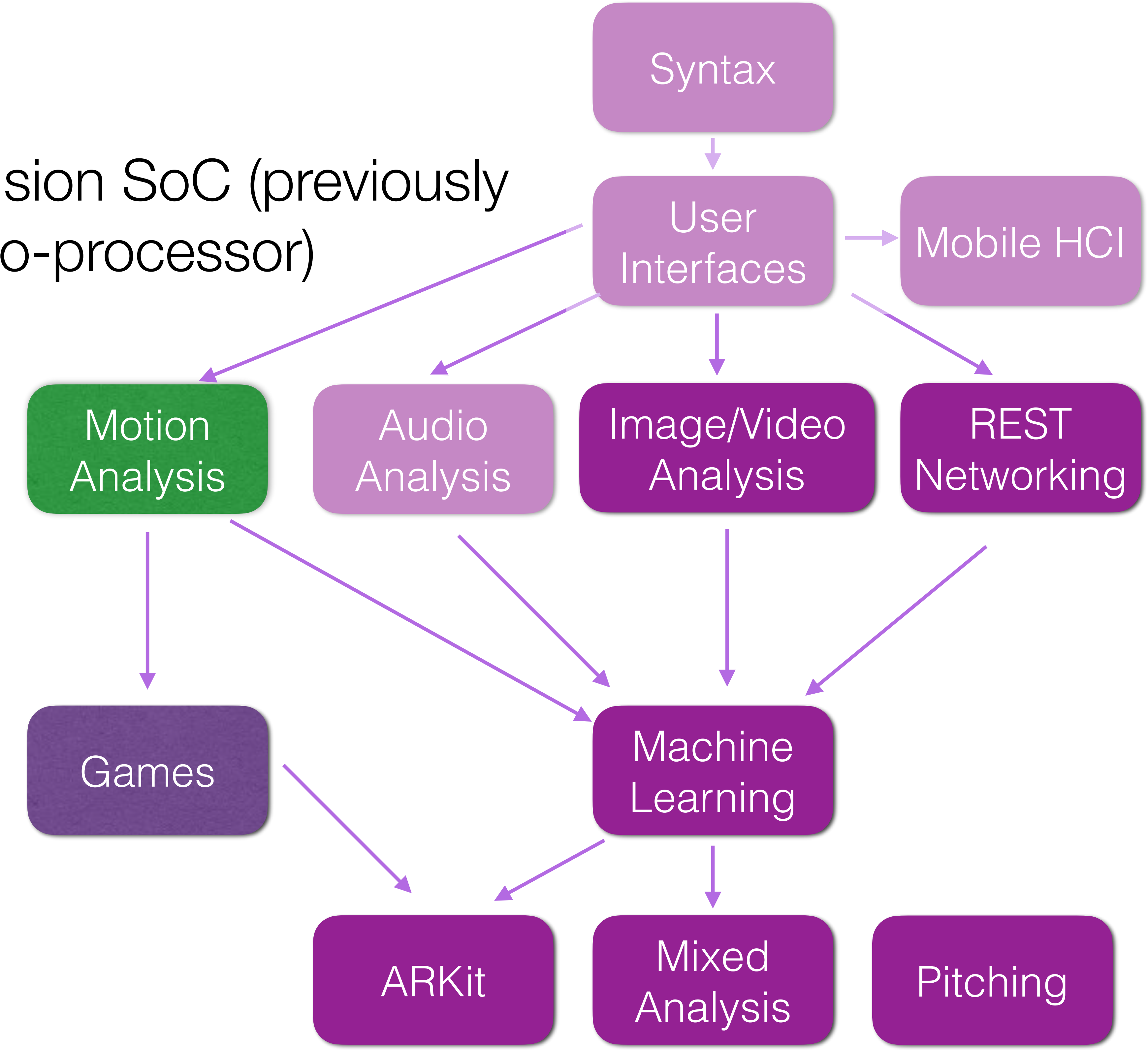


Questions on the FFT/audio

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

and now …

- core motion
  - A-series fusion SoC (previously M-series co-processor)

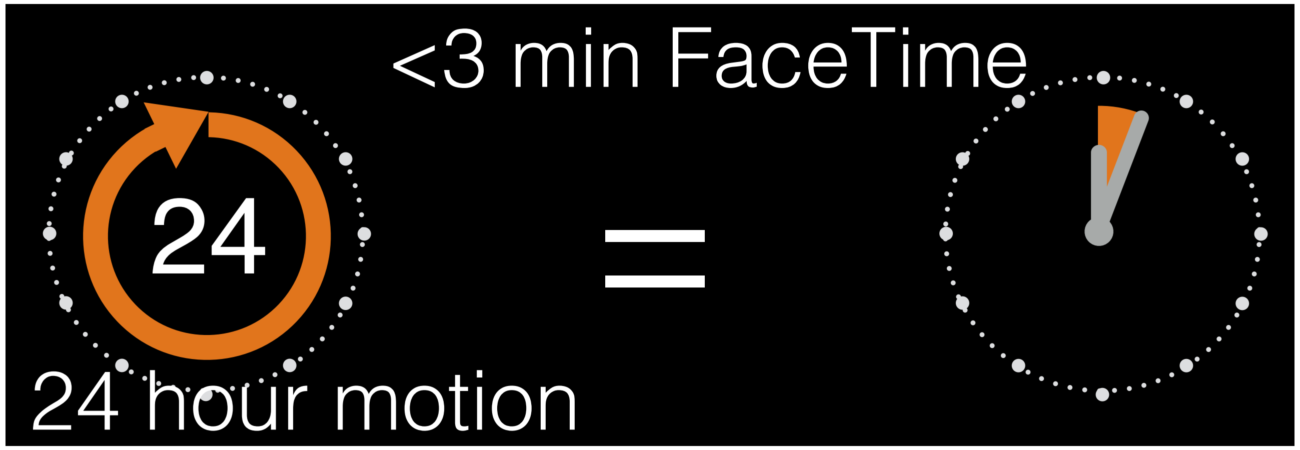


A-series fusion processor

- separate system on chip that reads all motion data from all “motion” sensors on the phone
    - accelerometer
    - magnetometer (compass)
    - gyroscope
    - barometer

- motion processor
- neural network engine
- GPU
- CPUs

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



<3 min FaceTime

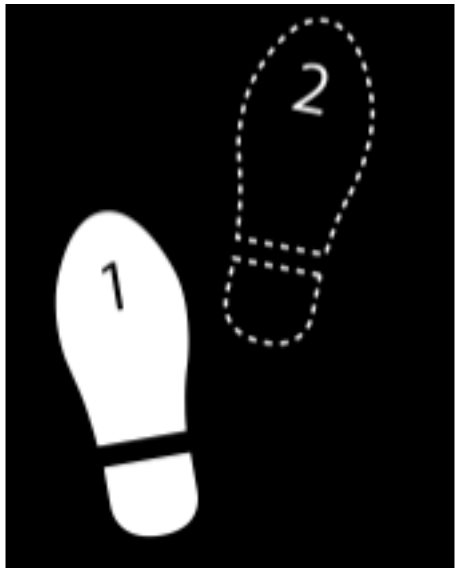
=

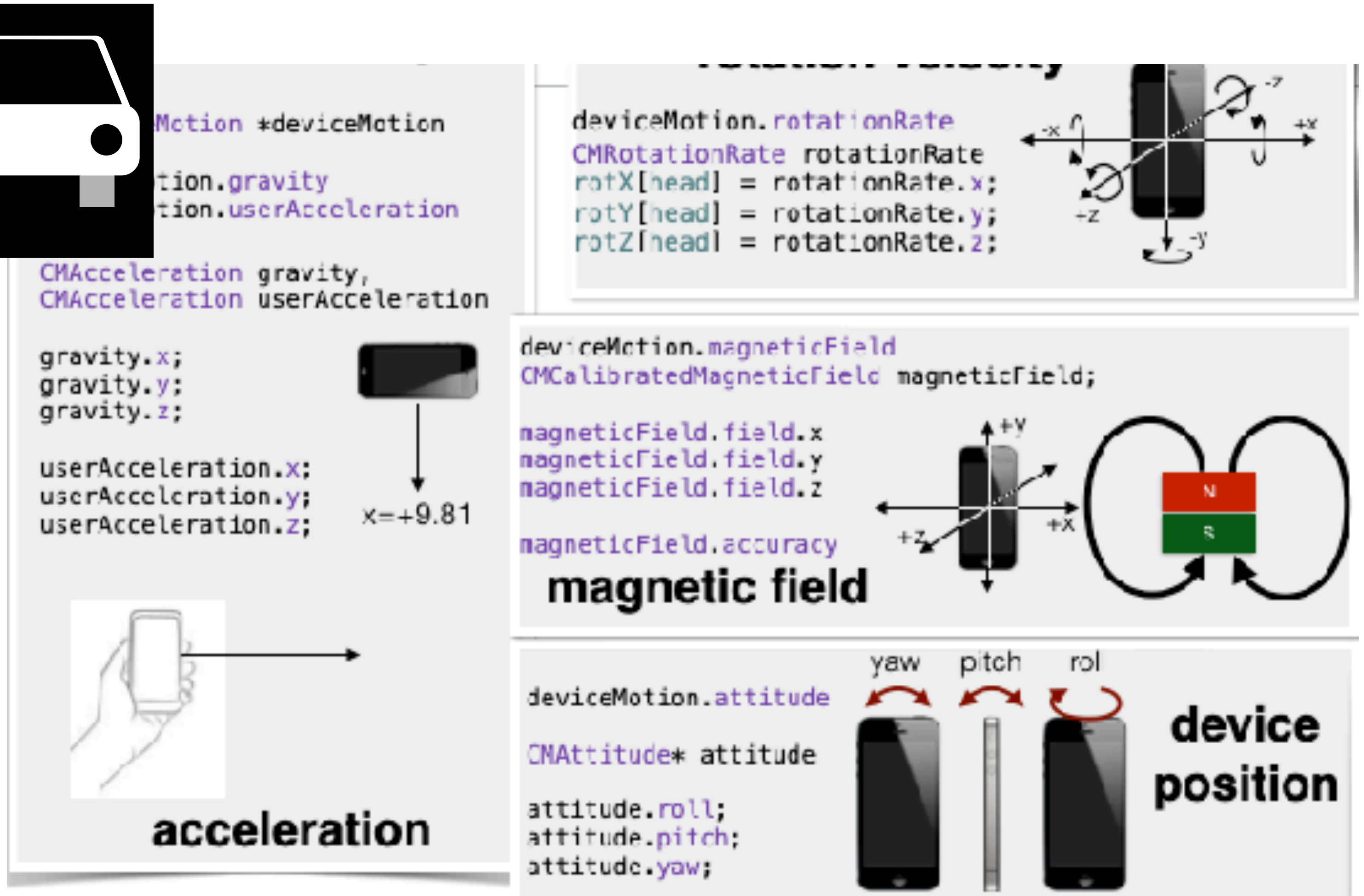
24 hour motion

- 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





```
Mction *deviceMotion
tion.gravity
tion.userAcceleration

CMAcceleration gravity,
CMAcceleration userAcceleration

gravity.x;
gravity.y;
gravity.z;

userAcceleration.x;
userAcceleration.y;
userAcceleration.z;
```

acceleration

```
deviceMotion.rotationRate
CMRotationRate rotationRate
rotX[head] = rotationRate.x;
rotY[head] = rotationRate.y;
rotZ[head] = rotationRate.z;
```

```
deviceMotion.magneticField
CMCalibratedMagneticField magneticField;

magneticField.field.x
magneticField.field.y
magneticField.field.z

magneticField.accuracy
```

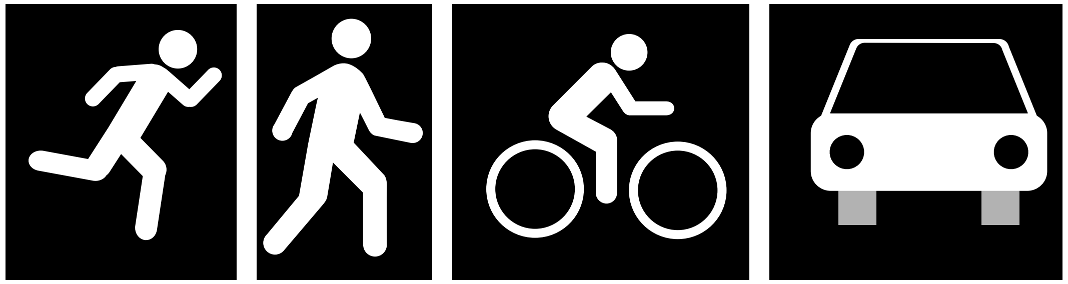
magnetic field

```
deviceMotion.attitude

CMAttitude* attitude

attitude.roll;
attitude.pitch;
attitude.yaw;
```

device position

high level streams

- not just raw data!
  - the A-fusion series 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 A-series

- 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**

subscribe to activity

```
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?

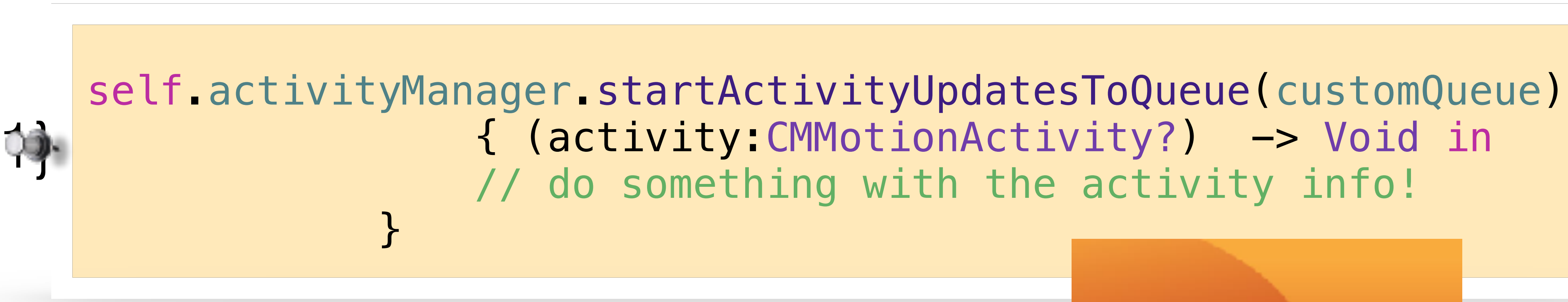
closure to handle updates
(*this one just prints description*)

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}

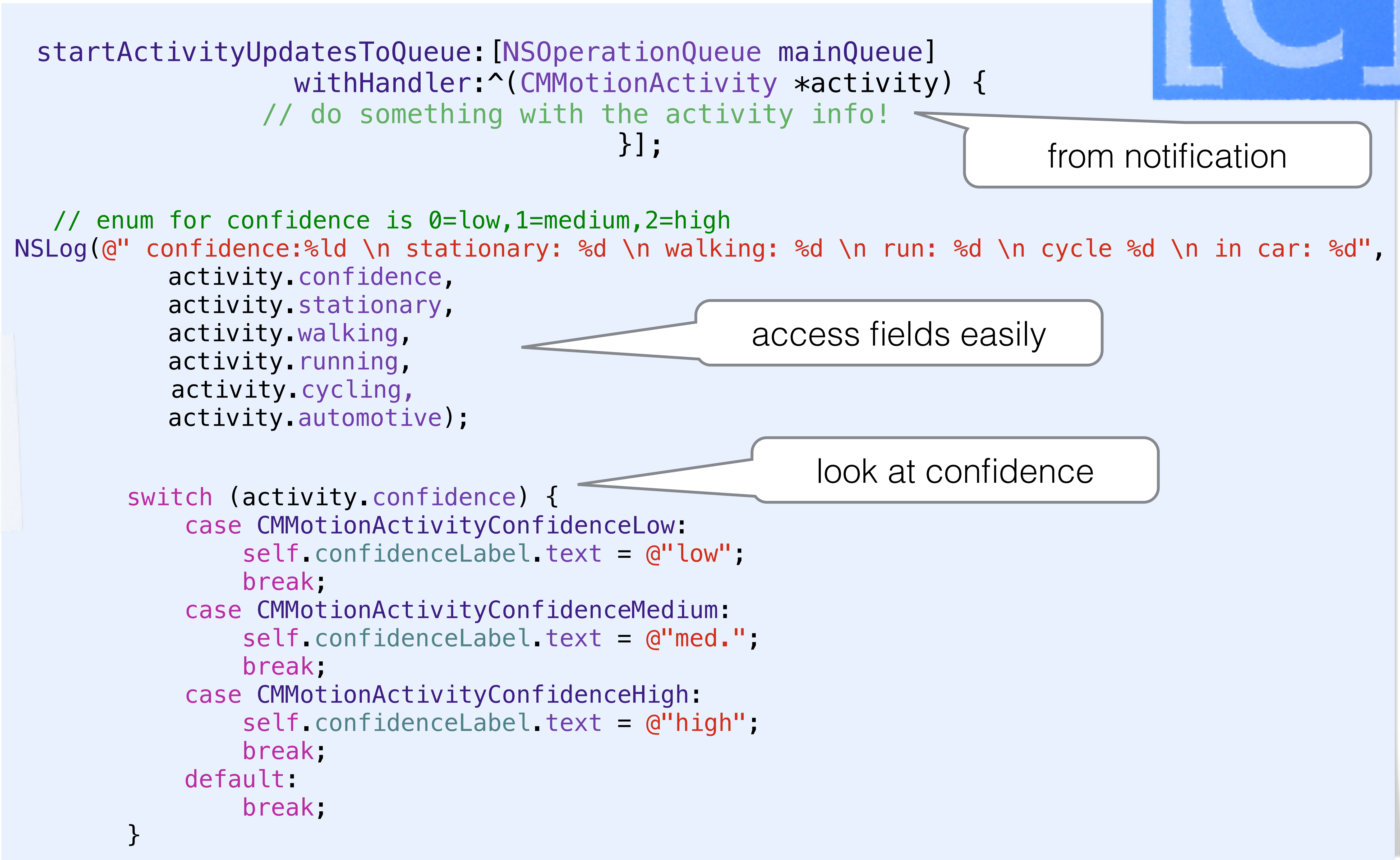
```
startActivityUpdatesToQueue:[NSOperationQueue mainQueue]
                withHandler:^(CMMotionActivity *activity)
{
              // do something with the activity info!
                                    }];
```



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

example update

inside
handler



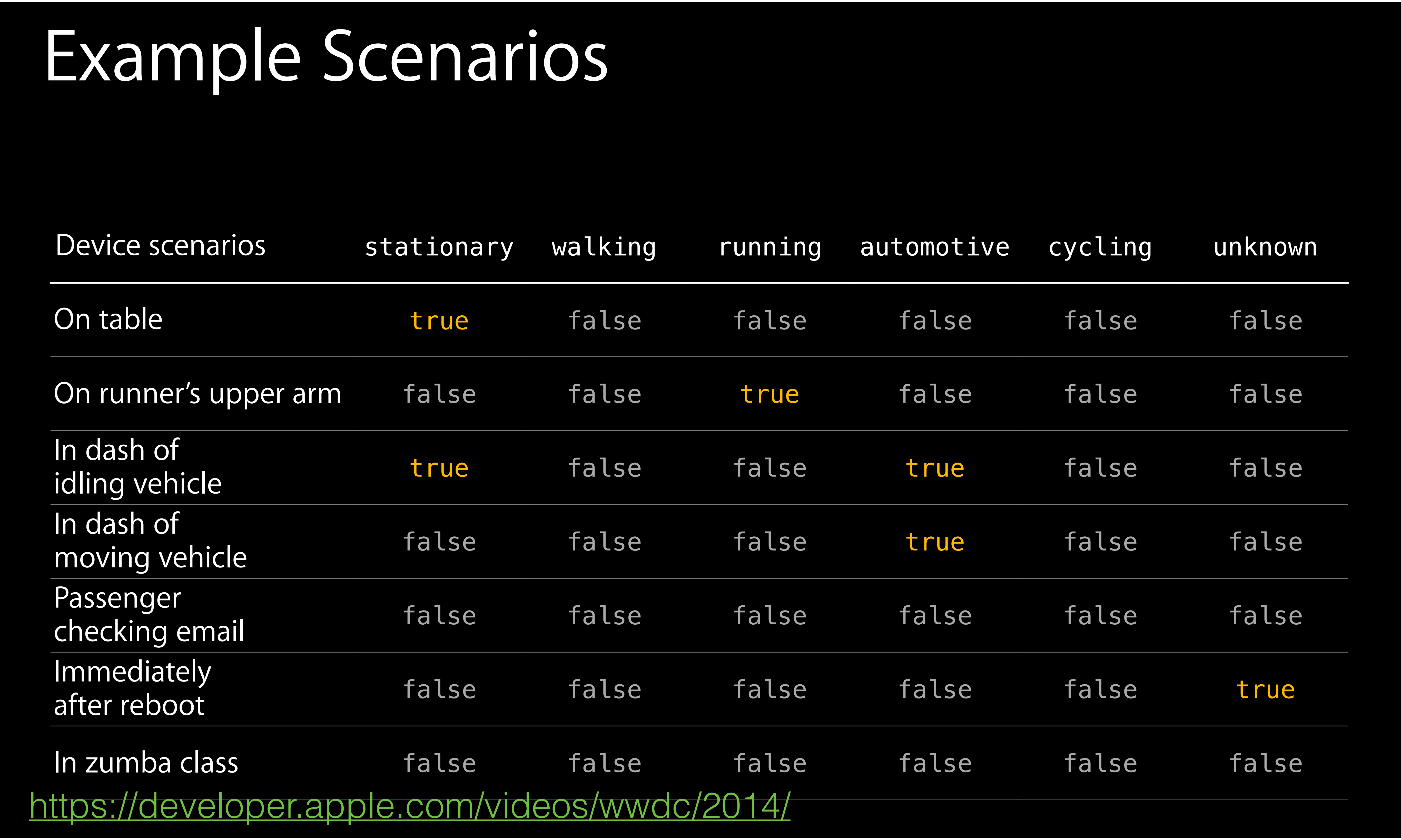
```
  startActivityUpdatesToQueue:[NSOperationQueue mainQueue]
                  withHandler:^(CMMotionActivity *activity) {
                // do something with the activity info!
                                      }];


  // 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?

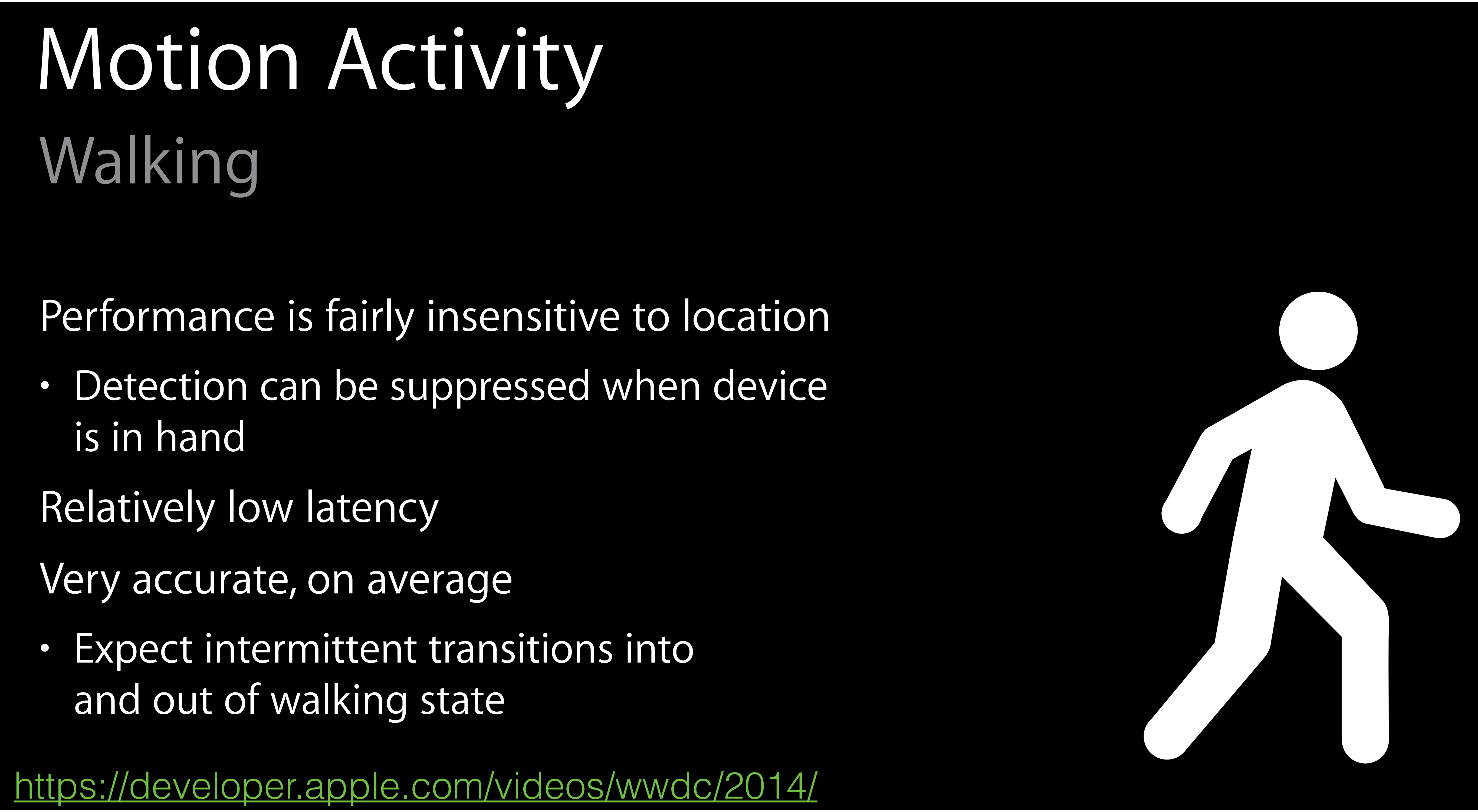


Example Scenarios

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/

what’s in an update?



Motion Activity

Walking

Performance is fairly insensitive to location

- Detection can be suppressed when device is in hand

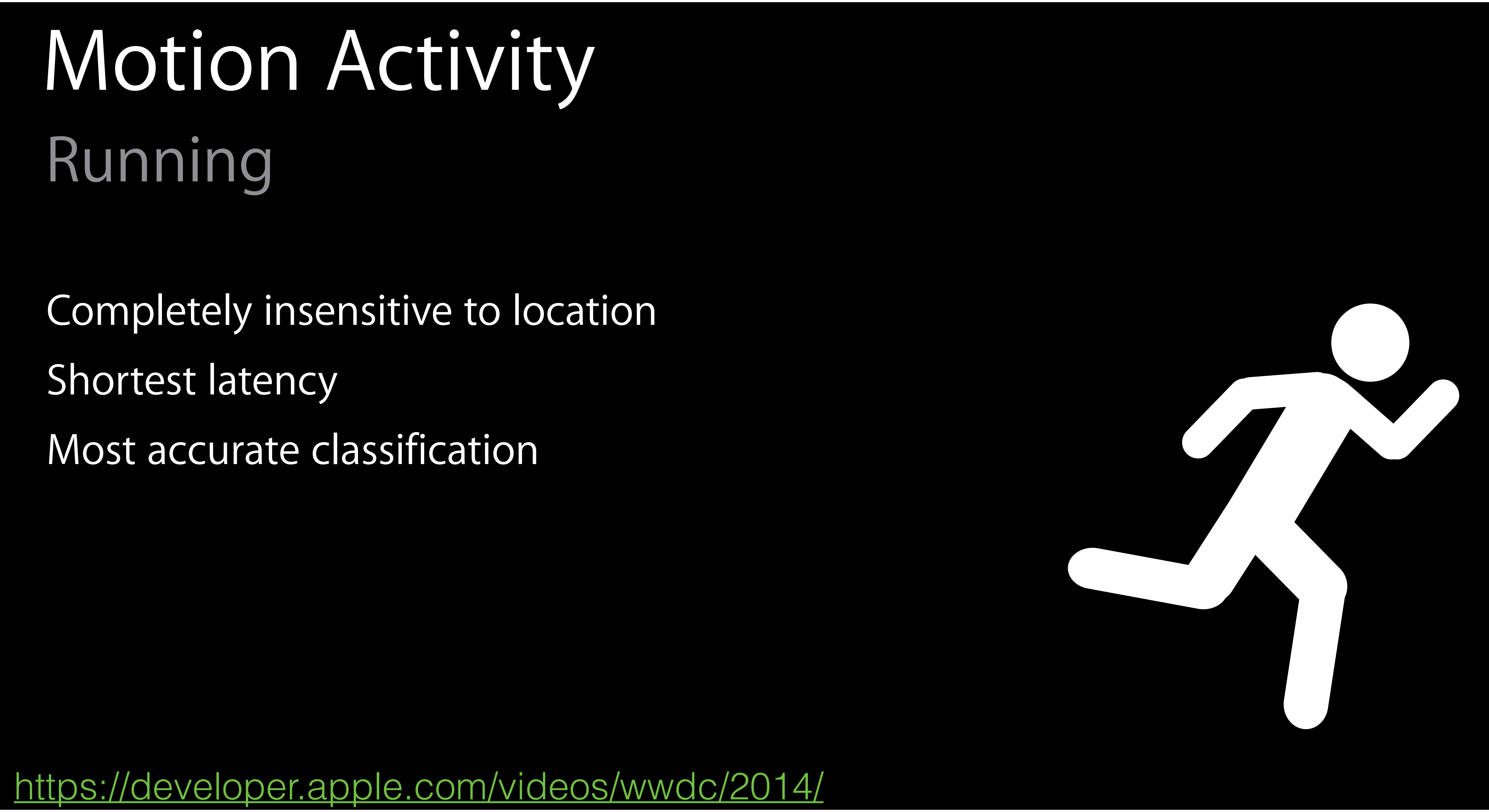
Relatively low latency

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

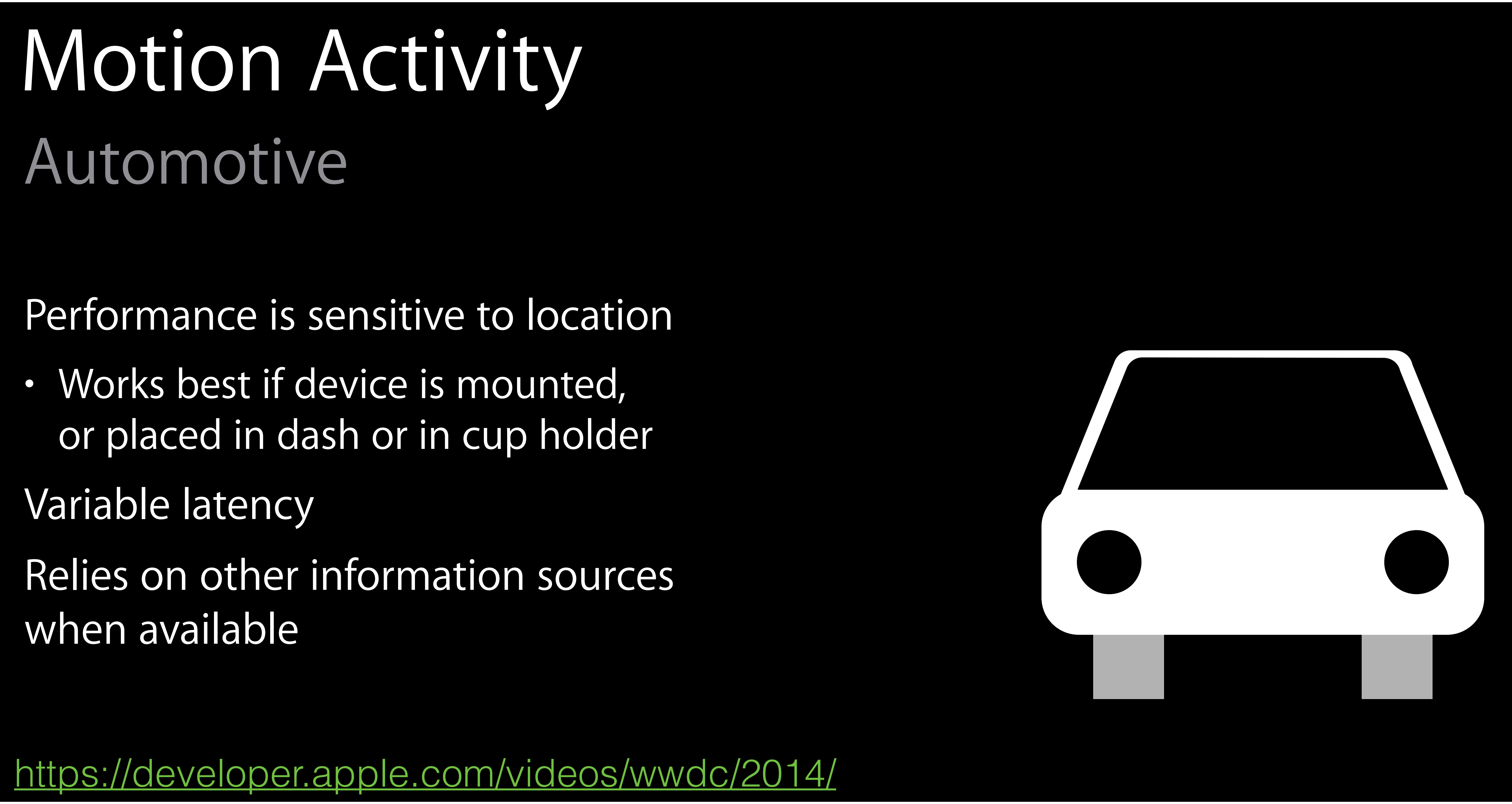
Completely insensitive to location

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

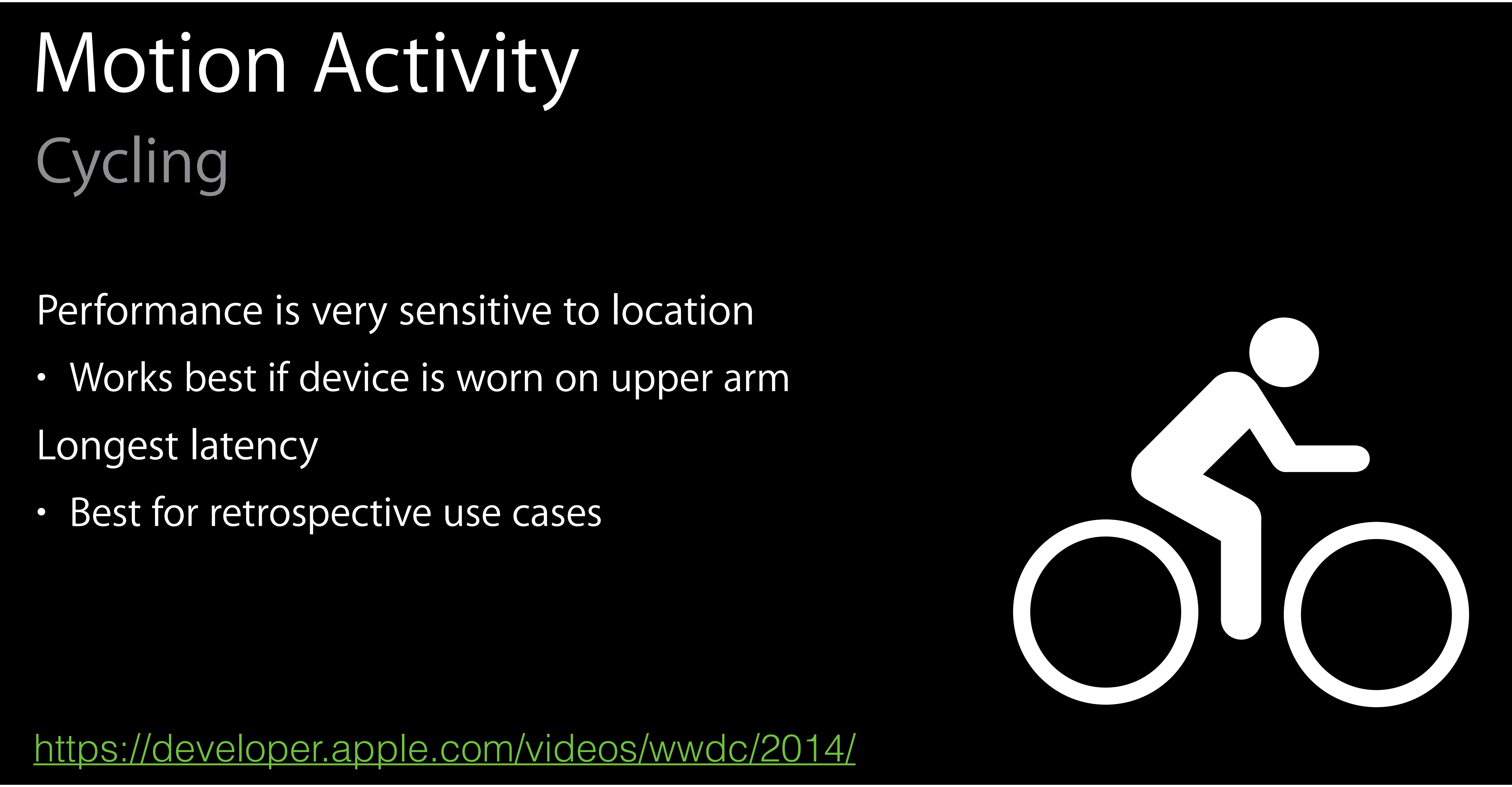
- Works best if device is mounted, or placed in dash or in cup holder

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

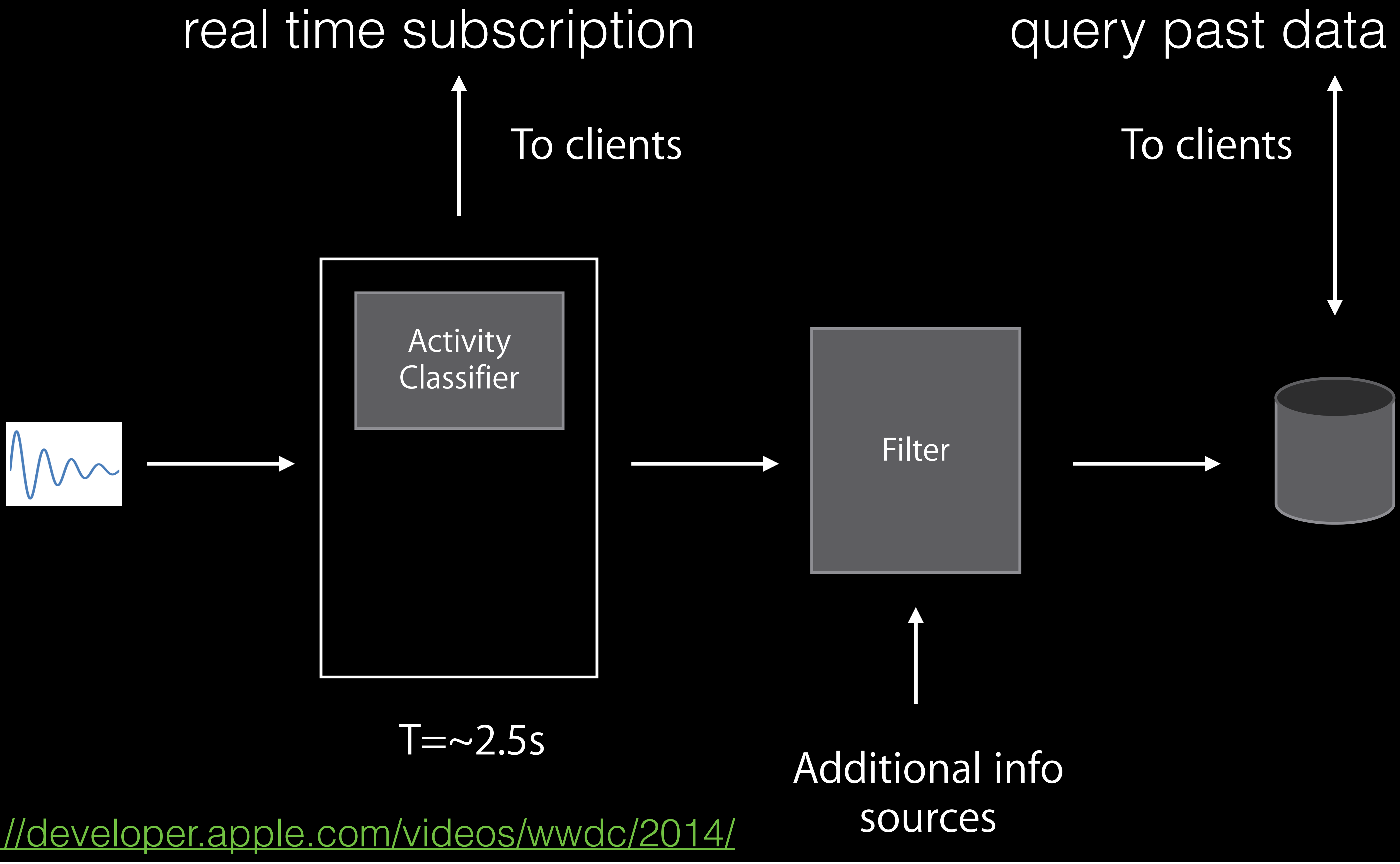
- Works best if device is worn on upper arm

Longest latency

- Best for retrospective use cases

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

Motion Processing Architecture



real time subscription

query past data

To clients

To clients

Activity Classifier

Filter

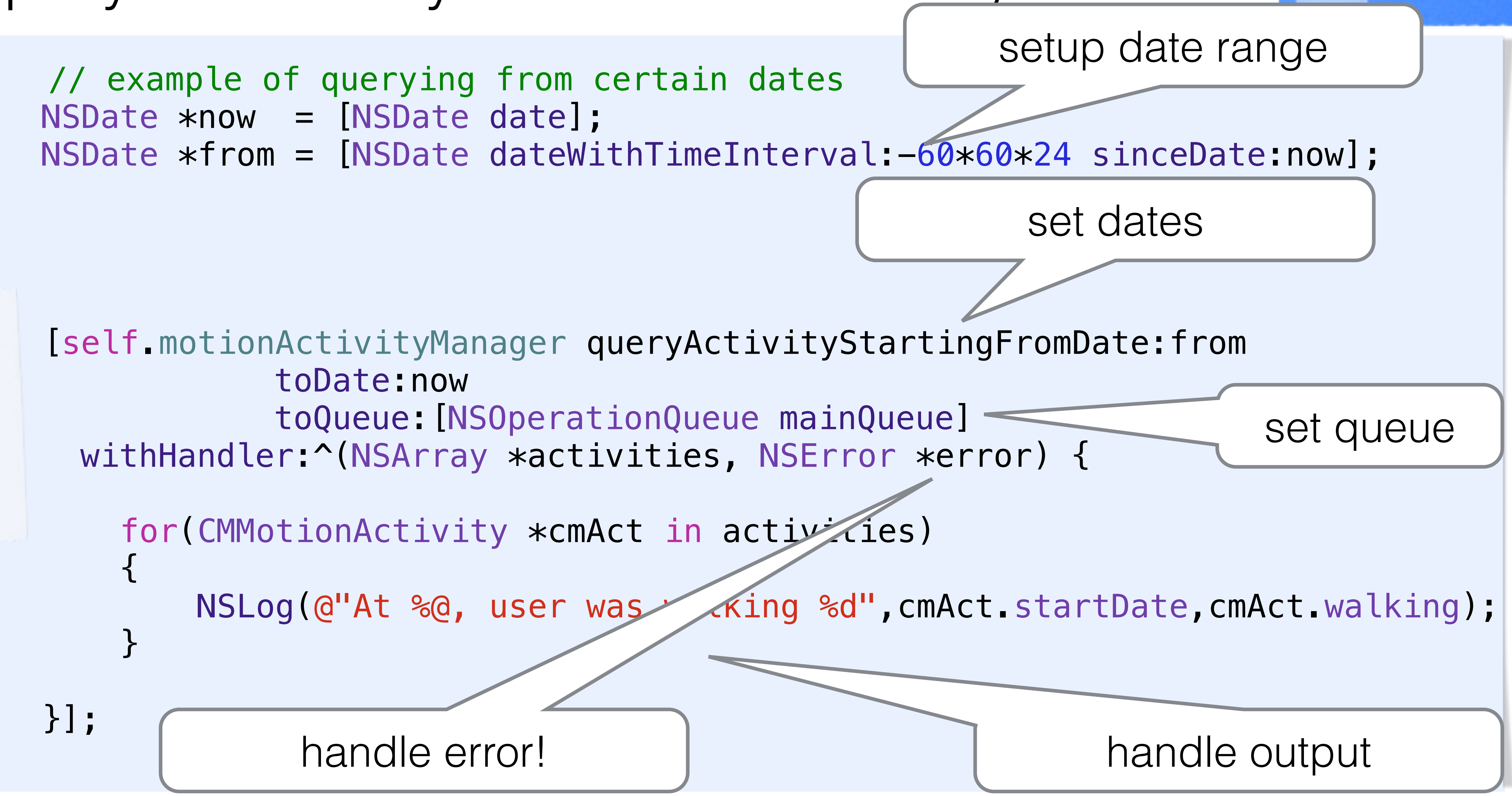
T=~2.5s

Additional info sources

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?

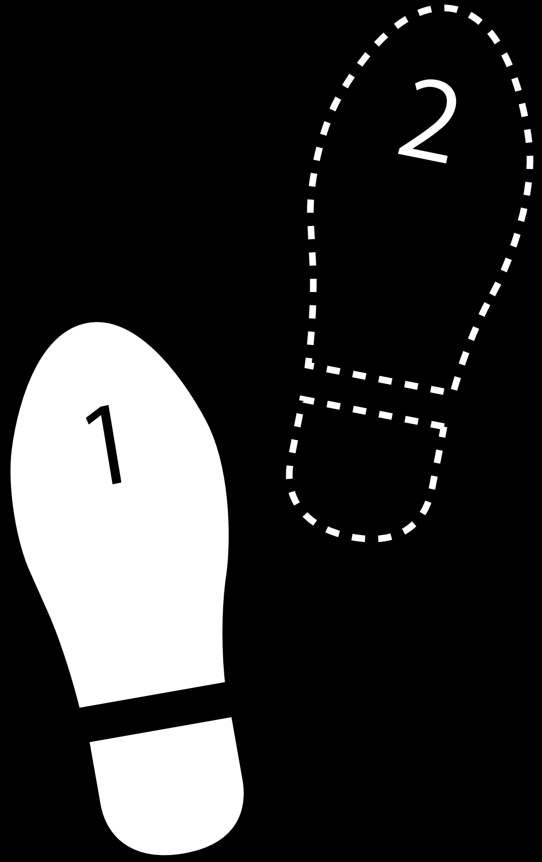
more than activity

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

- pedometer: special handling from the A-series?
  - CMPedometer

Pedometer

Step counting



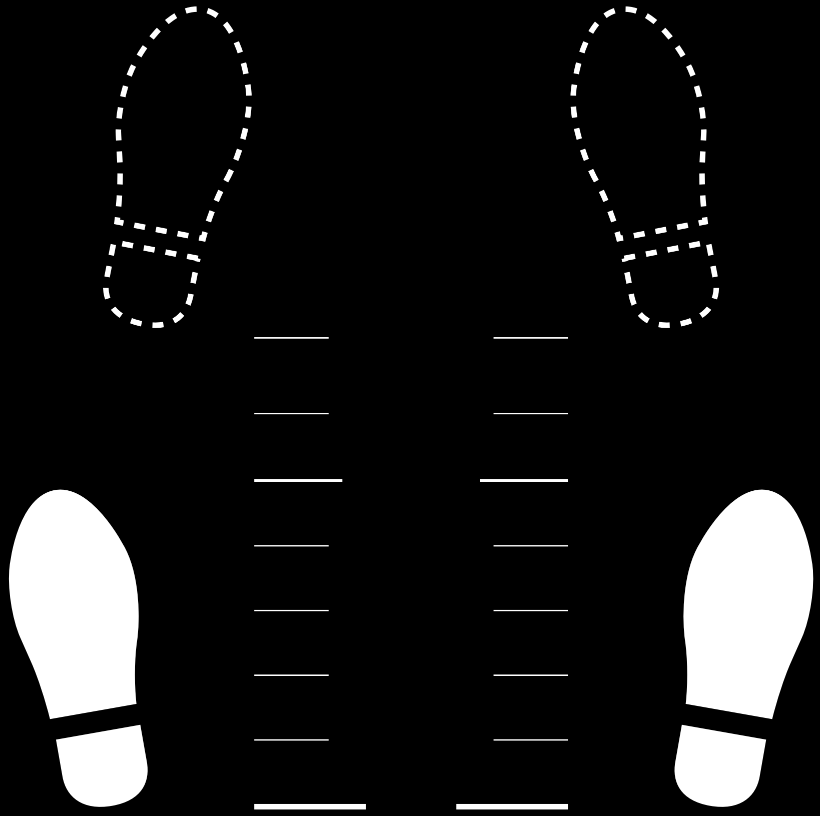
Consistent performance across body locations

Extremely accurate

Robust to extraneous motions

Pedometer

Stride estimation



Consistent performance across body locations

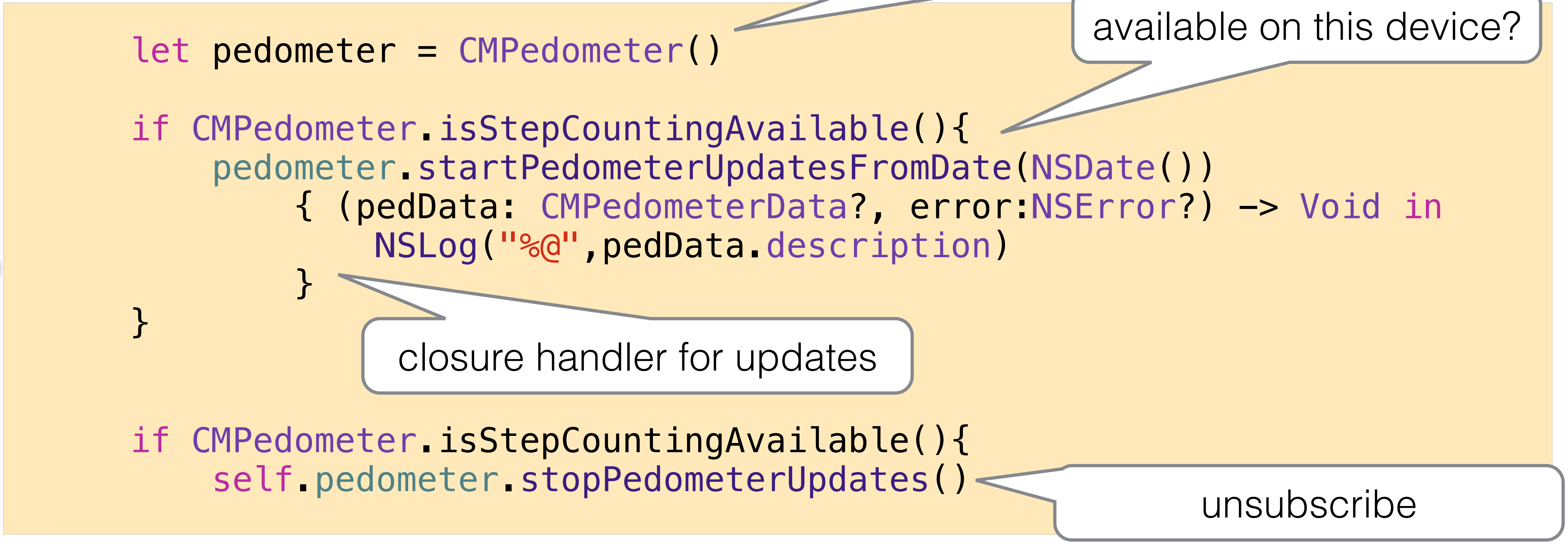
Consistent performance across pace

Extremely accurate

Adapts to the user over time

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

pedometer use



```
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

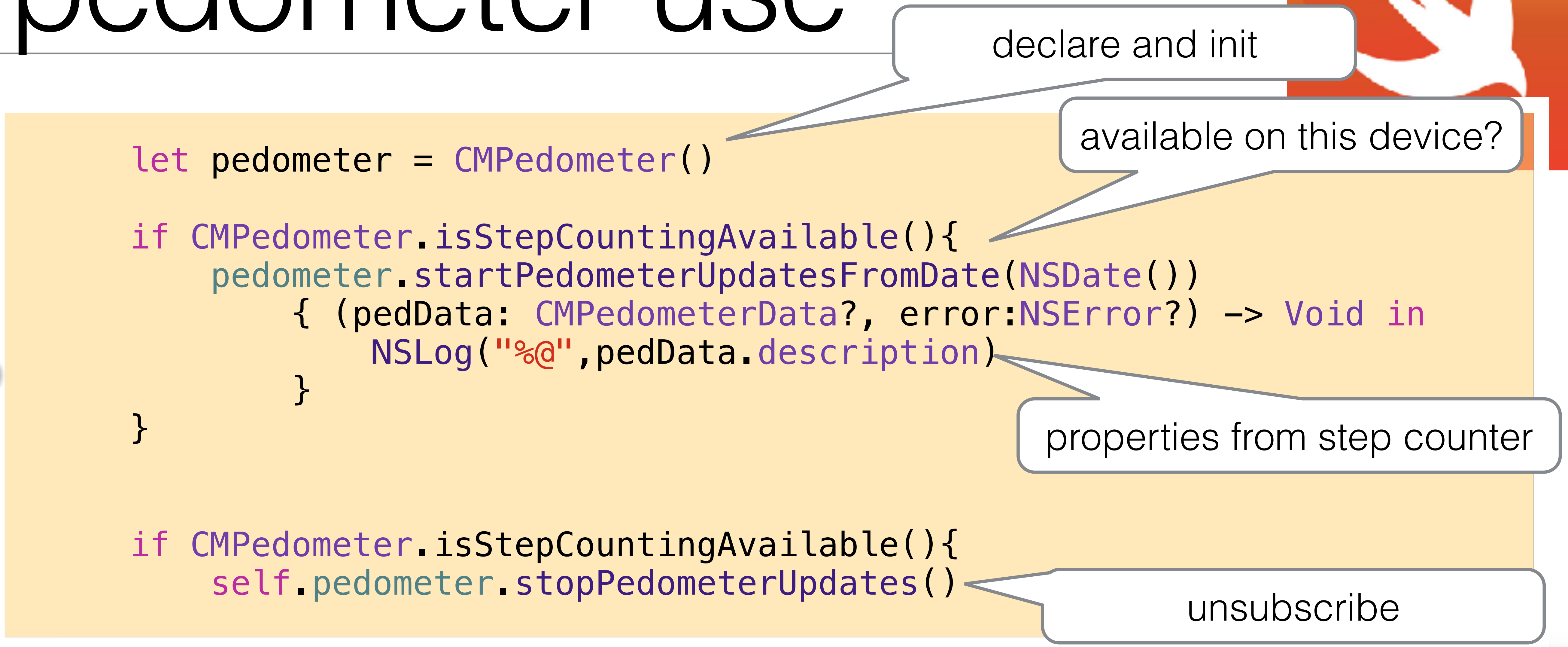
available on this device?

closure handler for updates

unsubscribe

pedometer use

revisiting



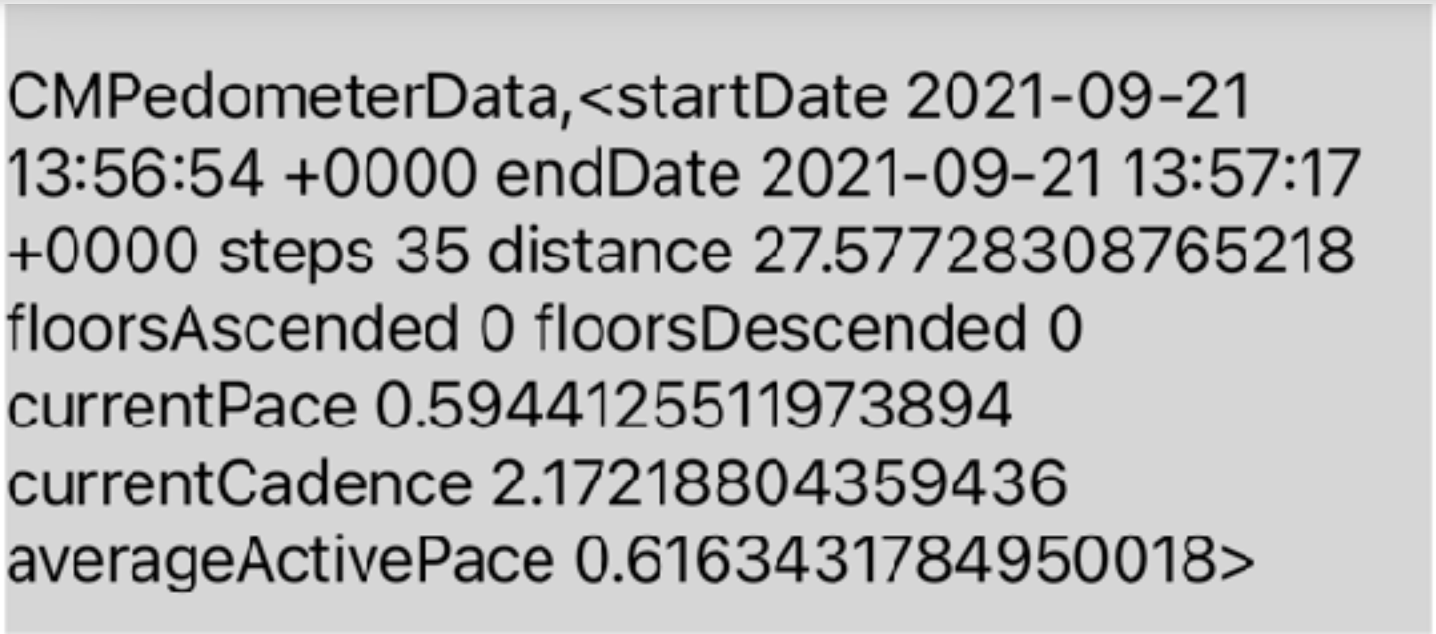
```
let pedometer = CMPedometer()

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 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 2.17218804359436
averageActivePace 0.6163431784950018>
```

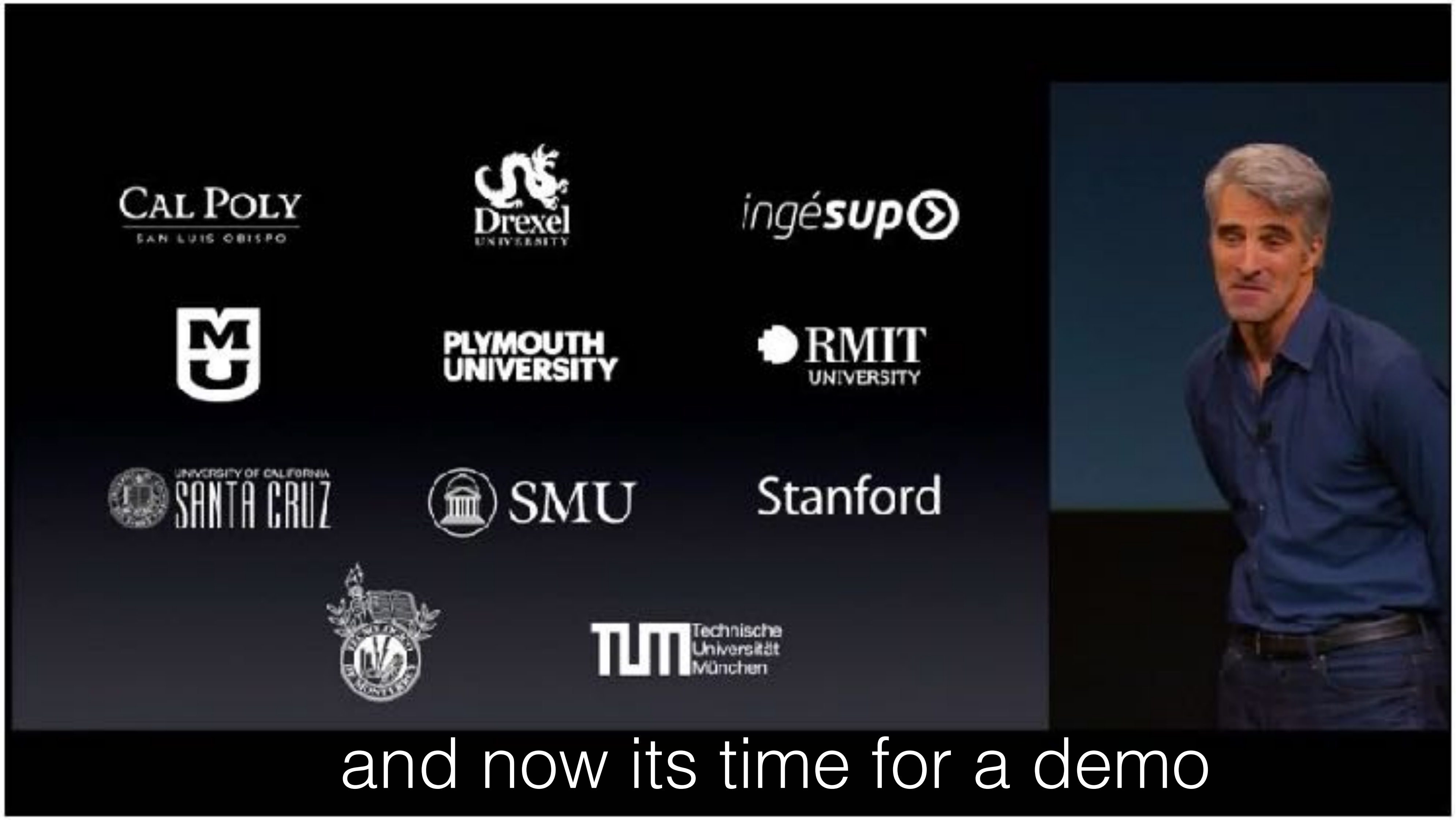
querying past steps

```
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

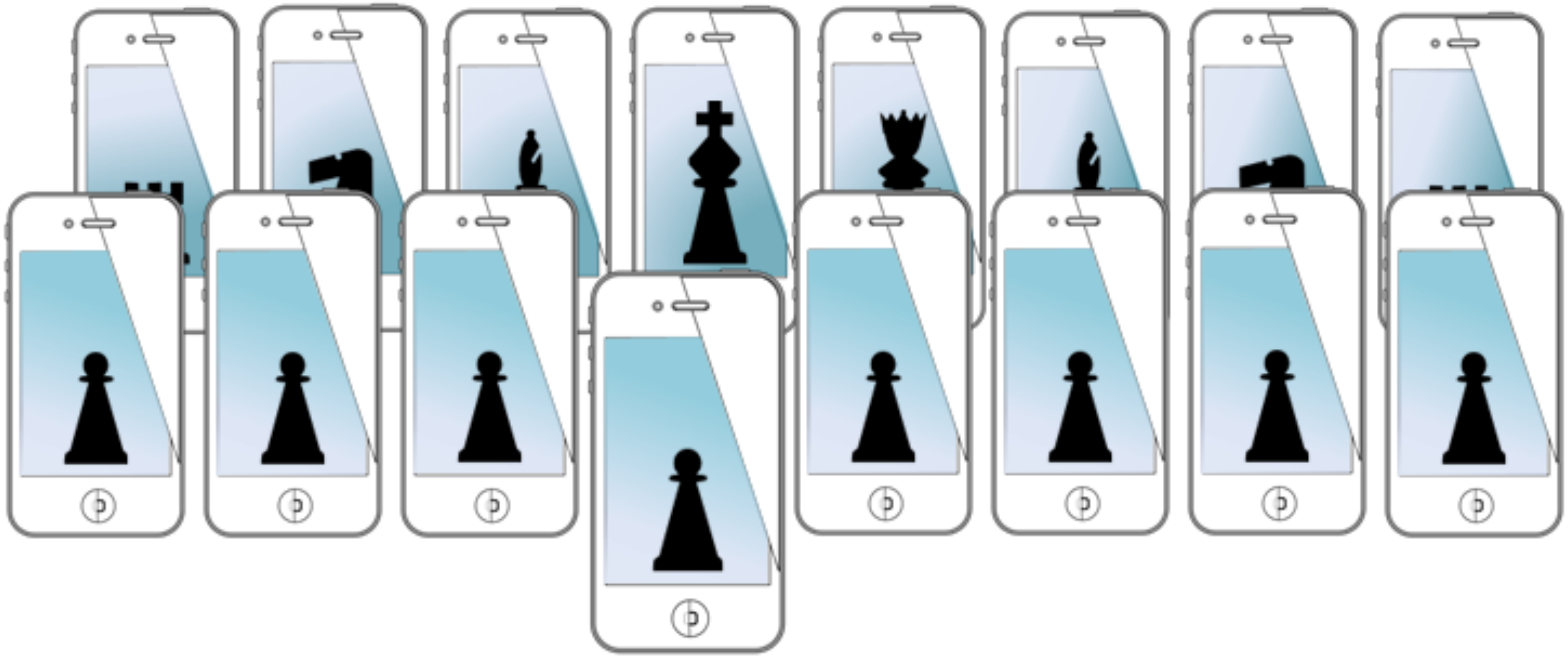
pedometer/activity demo



and now its time for a demo

if time!

MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

doppler and activity monitoring

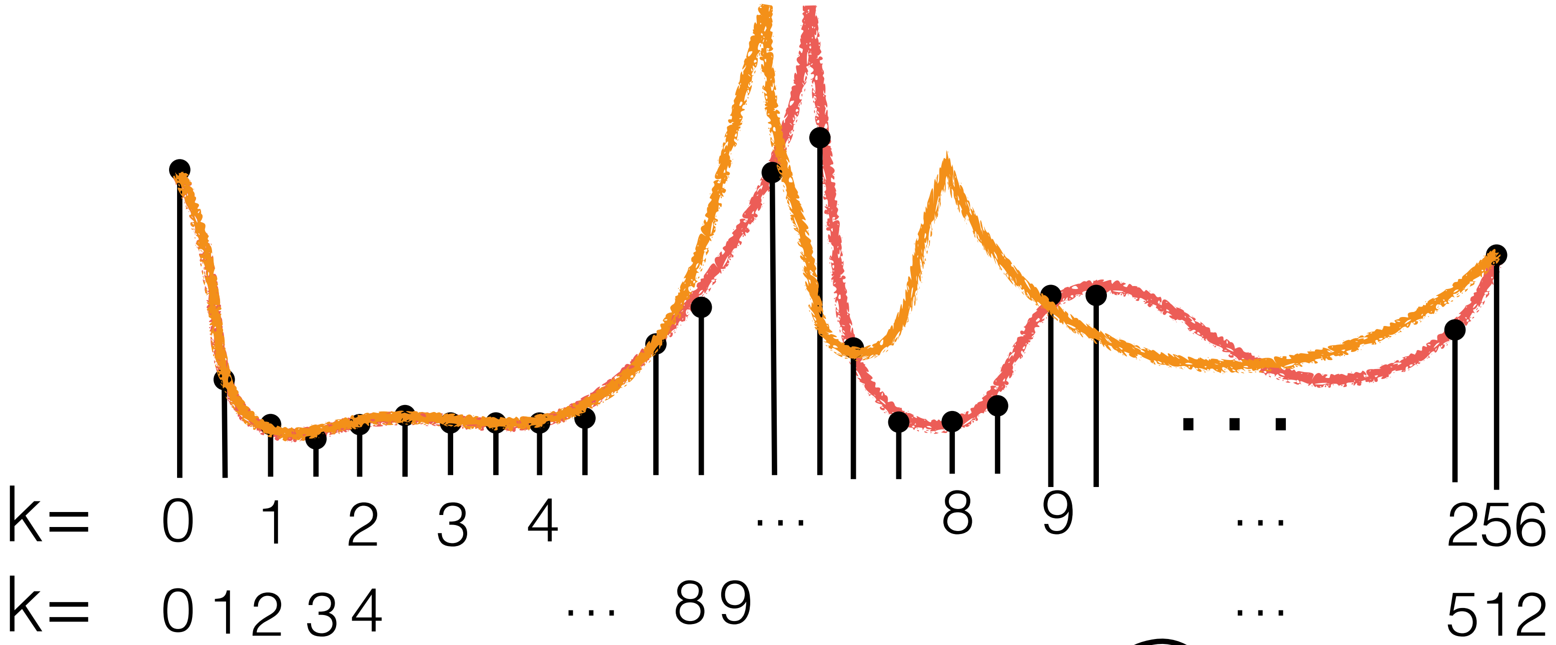
Eric C. Larson, Lyle School of Engineering,
Computer Science, Southern Methodist University

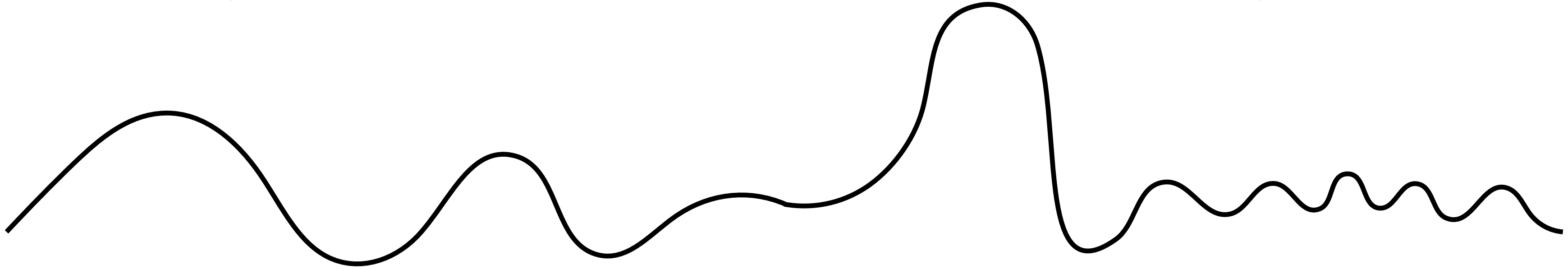
supplemental slides

- vector trajectory

resolution for the FFT

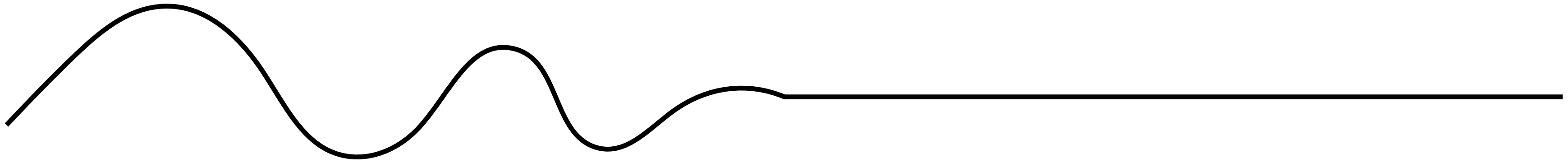
optional





256 points

next 256 points=512 total points



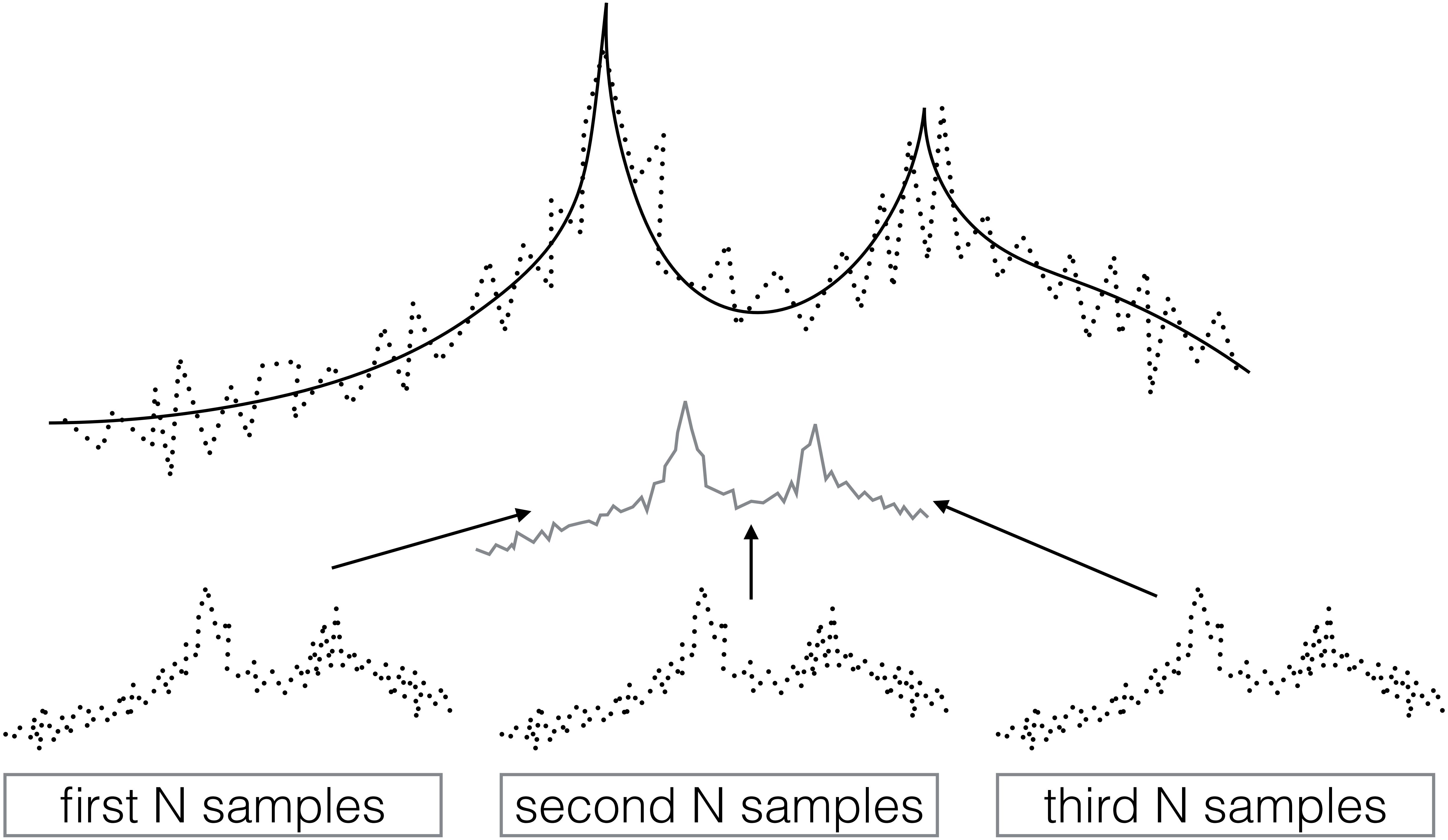
solution!

zero padding

noise in the FFT

optional

- variance around actual magnitude unavoidable



phone trajectory

- what direction is the phone (user) headed?
- direction could be:
  - cardinal {N, S, E, W}
  - altitude {sea level, +30 feet, etc.}
  - relative altitude {up, down}
  - relative trajectory {left, right, straight}
- how should we sense each of these?

GPS and magnetometer

GPS

motion sensors

motion sensors

up/down movement

- questions:
  - are we accelerating?
  - in what direction are we accelerating?
  - are we accelerating opposite of gravity?

which way is gravity?

```
deviceMotion.gravity.{x,y,z}
```

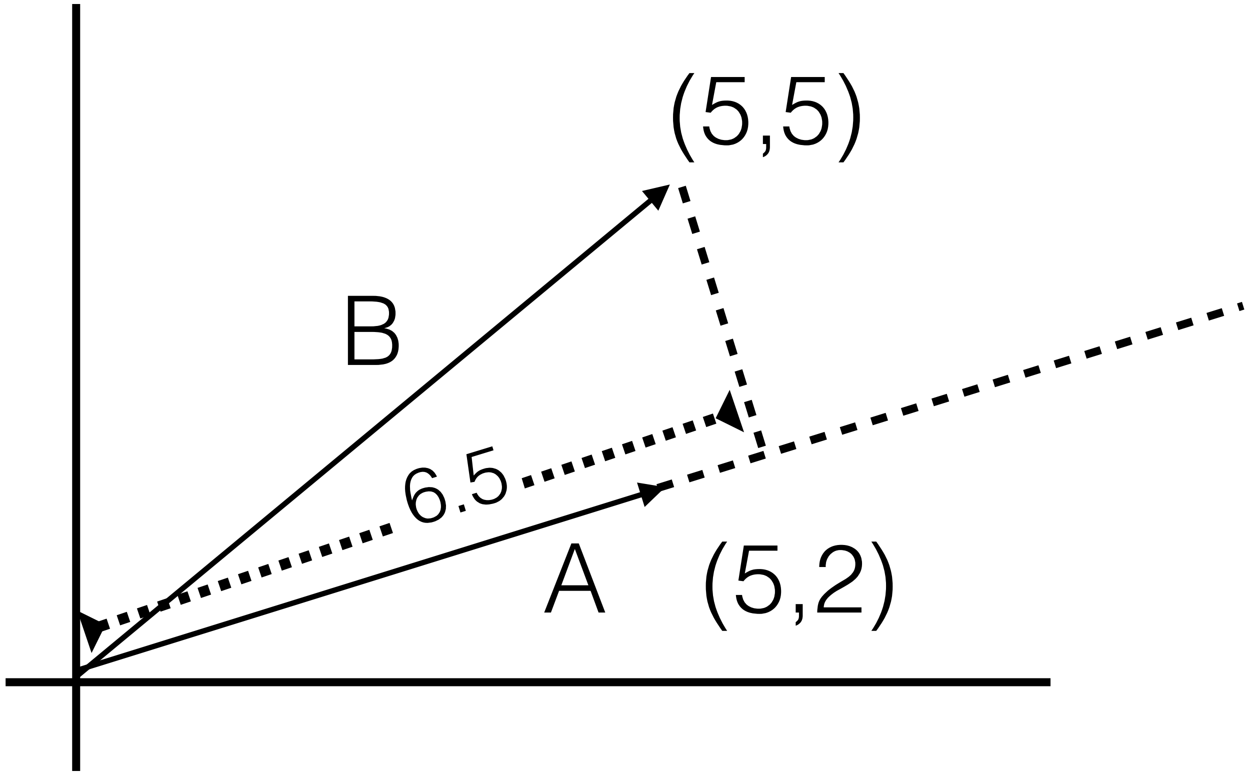
vectors

which way is the phone accelerating?

```
deviceMotion.userAcceleration.{x,y,z}
```

vector direction

- how much of one vector is in the direction of another?
- projections



$$\frac{A \bullet B}{|A|}$$

$$\frac{(5,5)\bullet(5,2)}{|(5,2)|}$$

$$\frac{5*5+5*2}{\sqrt{(5^2+2^2)}} = 35/\sqrt{29}$$

$$\sim 6.5$$

vector direction

- acceleration of the user towards or away from gravity?

```
CMAcceleration gravity, CMAcceleration userAccel

float dotProduct =
  gravity.x*userAccel.x + gravity.y*userAccel.y + gravity.z*userAccel.z;

float normDotProd =
  dotProduct / (gravity.x*gravity.x + gravity.y*gravity.y + gravity.z*gravity.z);
```

positive acceleration is speeding up
negative acceleration is slowing down

vector acceleration demo

- don’t drop it!

profiling demo

- using the instruments panel in Xcode
  - memory leaks
  - general efficiency
  - excellent integration with iOS