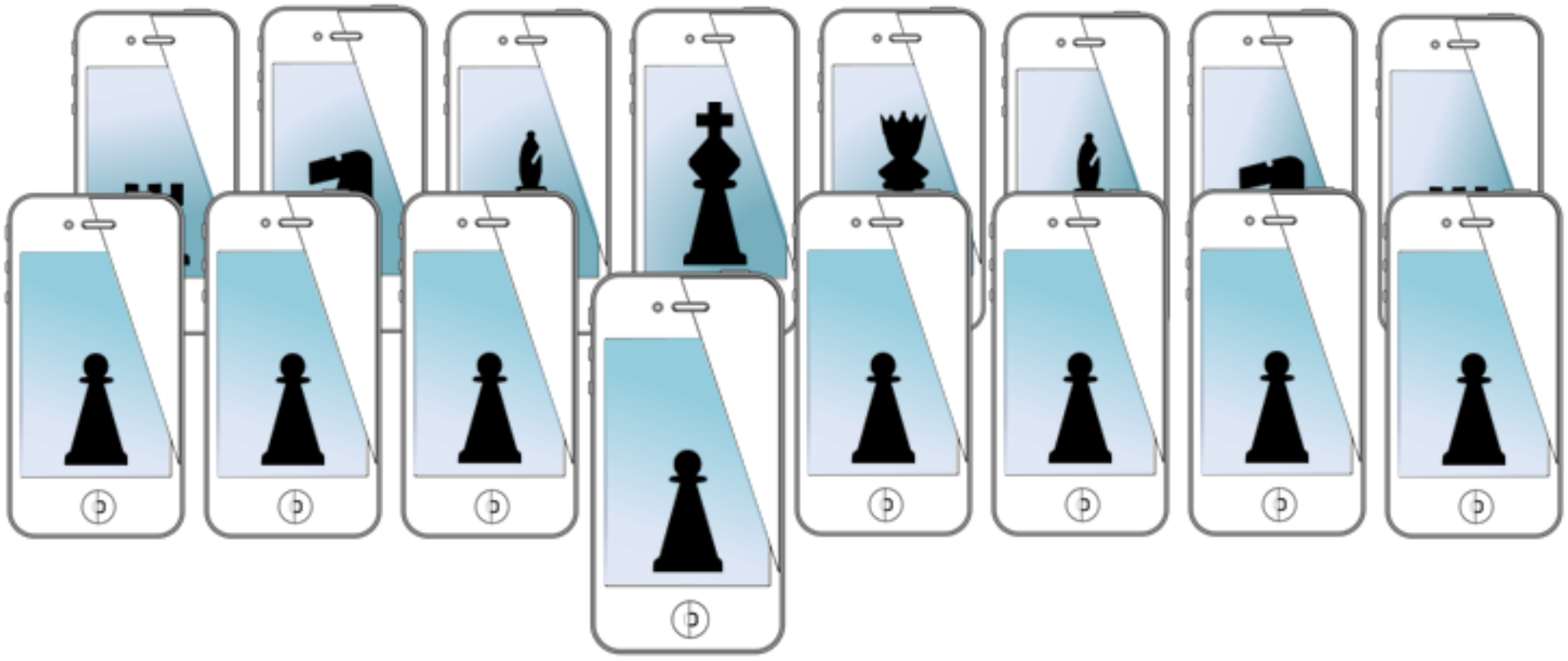
MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

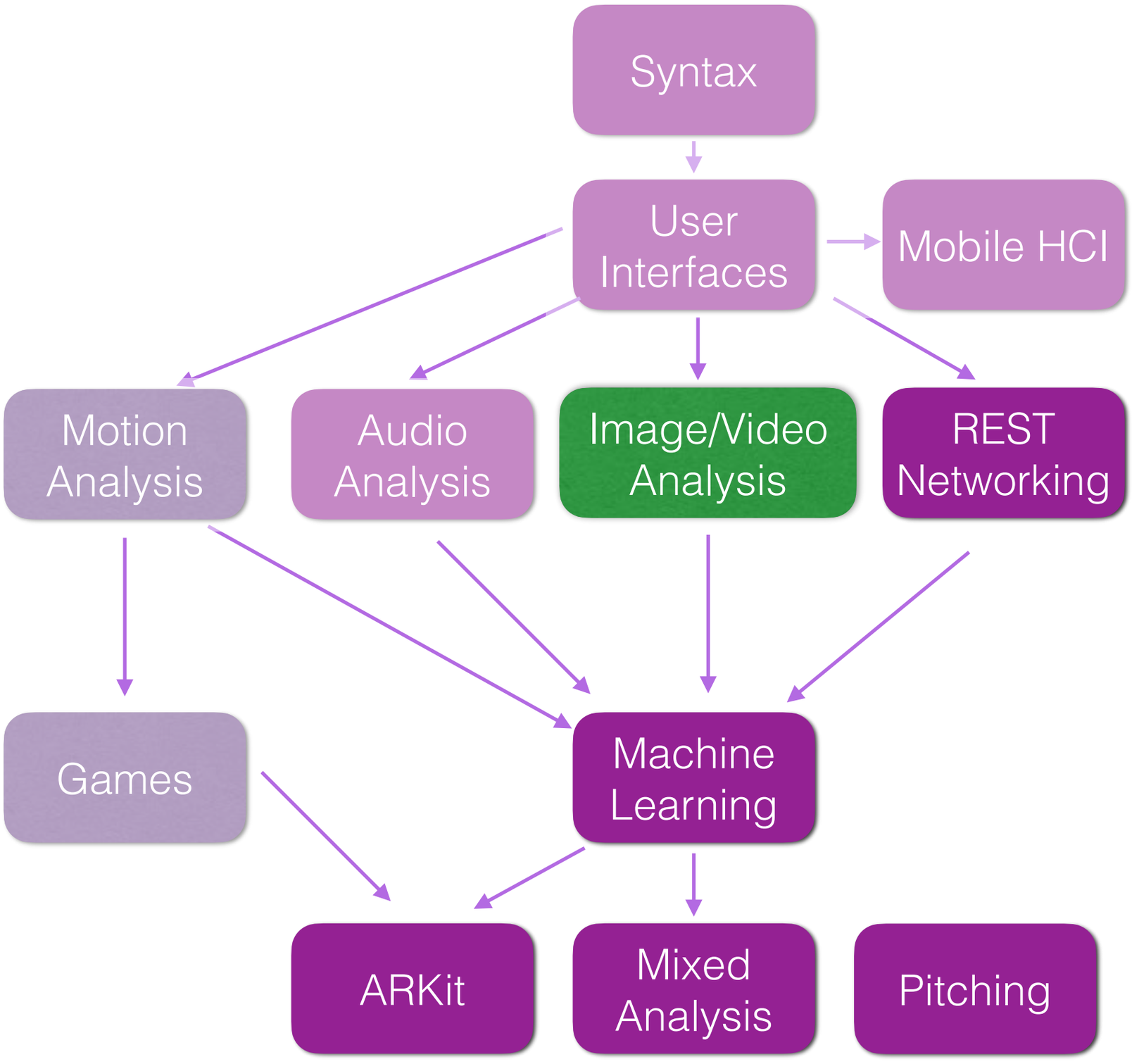
computer vision with core image

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

agenda

- Logistics:
  - grading update
- Agenda:
  - video processing
    - computer vision
      - face detection
      - heart physiology (optional)
      - introduction to the vision API
- If we finish lecture material for video processing…

class overview



last time: core image demo

- ImageLab, using the camera via delegation

but we want video…

- want to access incoming video in real time
  - and display to screen!
- that is a lot of processing
- setup needs to occur in conjunction with GPU
  - setup proper context (i.e., OpenGL)
  - renderers, processing pipeline
- you need to understand this intuitively
  - but you won't write the code to do it

AVCaptureSession

- mediates all access to incoming video and view
- the screen output and camera input need to speak the same language
  - same color representation (BGRA vs ARGB)
  - and transforms (mirroring, rotation)
  - important: same rendering context (for speed)
- capture session is optimized for video chat
  - so audio can also be captured here (not unlike Novocaine)

AVCaptureSession

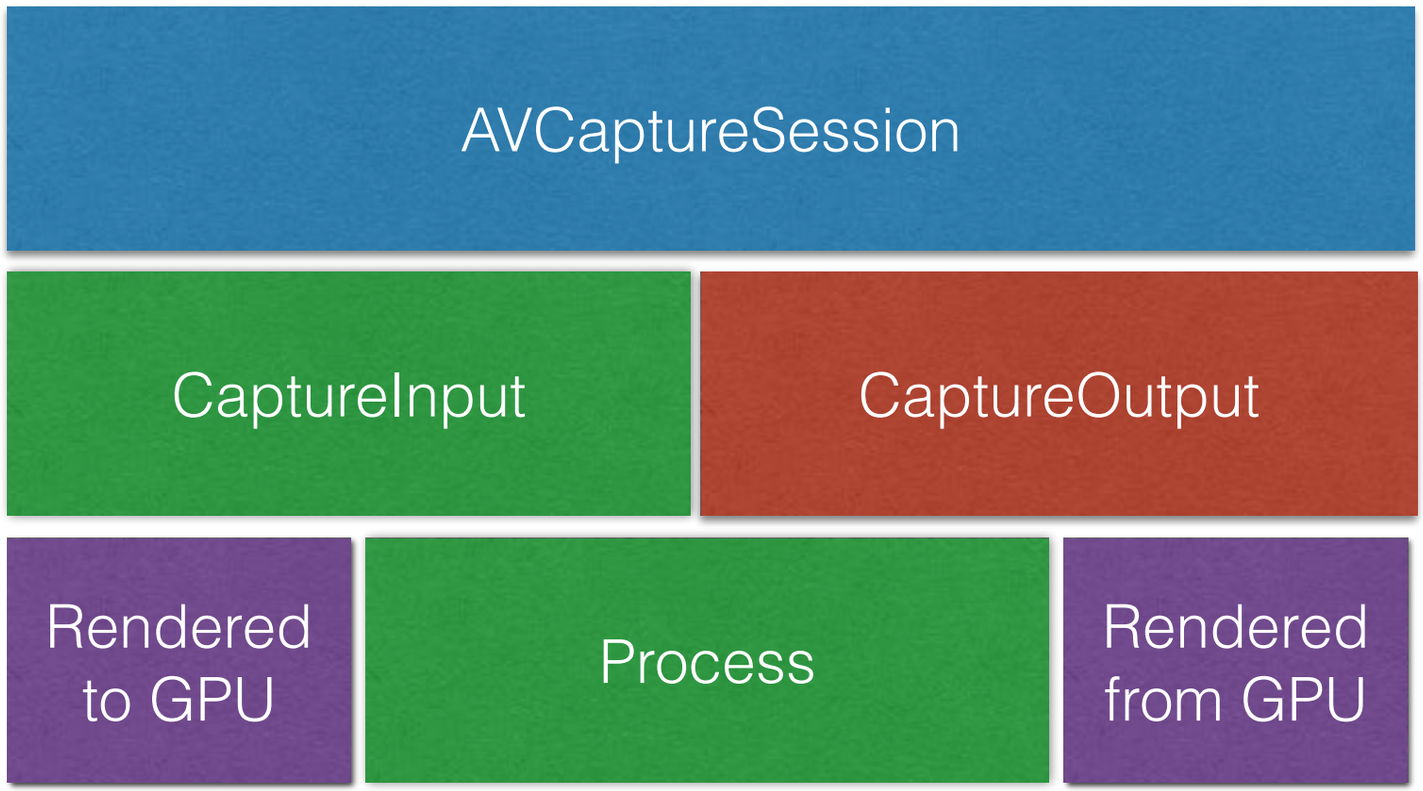
- setup the capture
  - device: front or back camera

```
AVCaptureDevicePositionFront
AVCaptureDevicePositionBack
```

  - quality preset:

```
NSString *const AVCaptureSessionPresetPhoto;
NSString *const AVCaptureSessionPresetHigh;
NSString *const AVCaptureSessionPresetMedium;
NSString *const AVCaptureSessionPresetLow;
NSString *const AVCaptureSessionPreset352x288;
NSString *const AVCaptureSessionPreset640x480;
NSString *const AVCaptureSessionPreset1280x720;
NSString *const AVCaptureSessionPreset1920x1080;
NSString *const AVCaptureSessionPresetiFrame960x540;
NSString *const AVCaptureSessionPresetiFrame1280x720;
```

conceptual architecture



use core image, with processing setup for GPU
no data transfer from the GPU!

VisionAnalgesic

- what did we do with audio?
  - don't reinvent the wheel: use Novocaine
- now: use **VisionAnalgesic.swift**
  - takes the pain out of GPU capture, render, and processing

declare

init

start

options

stop

```
var visionAnalgesic:VisionAnalgesic! = nil


self.visionAnalgesic = VisionAnalgesic(view:self.cameraView)

if !self.visionAnalgesic.isRunning{
    self.self.visionAnalgesic.start()
}

self.visionAnalgesic.setPreset(AVCaptureSession.Preset.medium)
self.visionAnalgesic.toggleCameraPosition()
self.visionAnalgesic.setCameraPosition(AVCaptureDevice.Position.front)


if self.visionAnalgesic.isRunning{
            self.visionAnalgesic.stop()
            self.visionAnalgesic.shutdown()
        }
```

Must be an MTKView

VisionAnalgesic

- processing: similar to Novocaine
  - assumed that the output is always the screen of phone
  - use blocks and return image to draw to screen

```
        // setup a block to perform any processing
         self.visionAnalgesic.setProcessingBlock()
             {(inputImage:CIImage)->(CIImage) in
                 return inputImage
         }
```

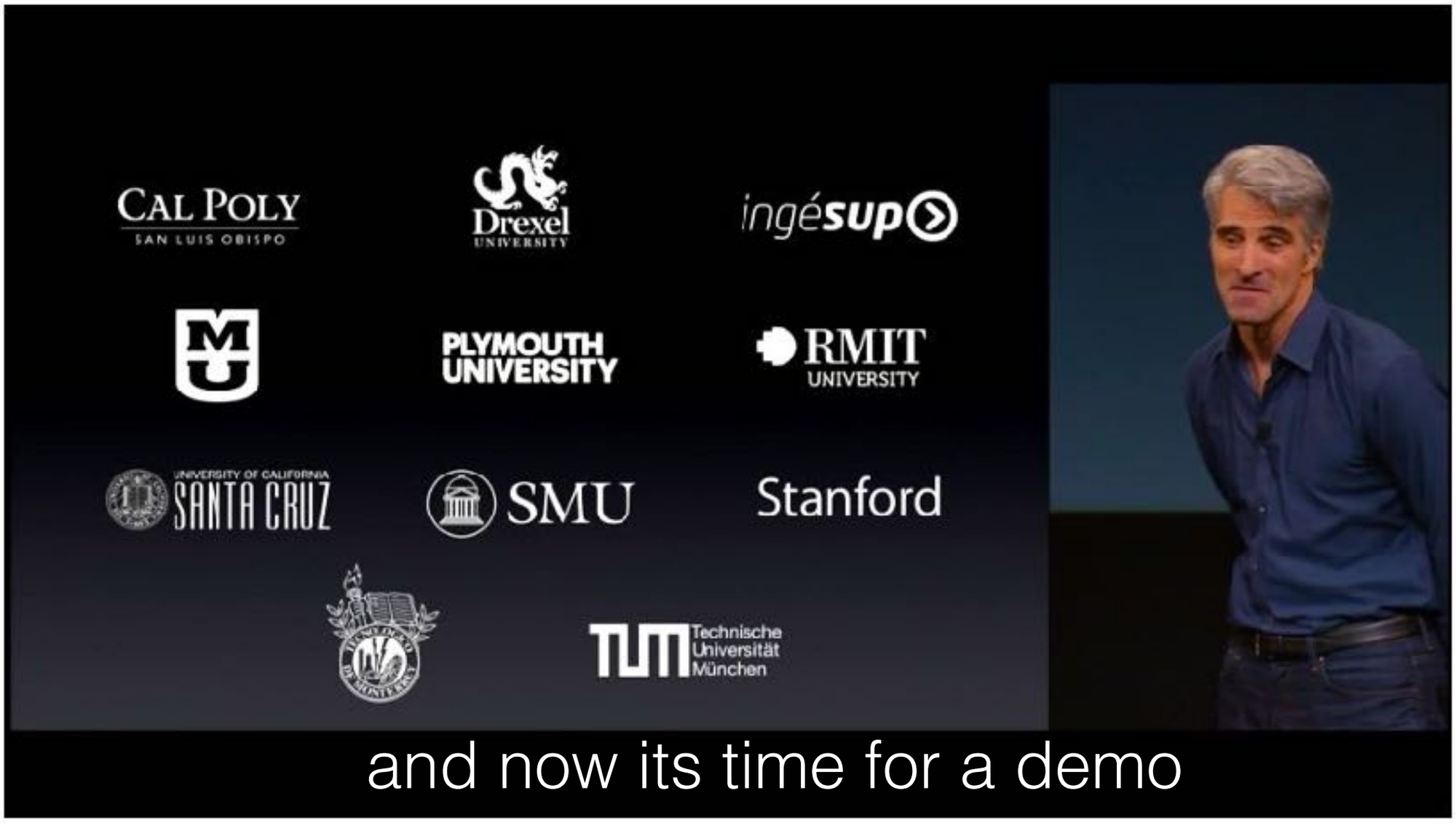
image from camera passed in

return image to draw to screen

```
    let filter:CIFilter = CIFilter(name: "CIBloom")

        // setup a block to perform any processing
        self.visionAnalgesic.setProcessingBlock()
            {(inputImage:CIImage)->(CIImage) in
                filter.setValue(inputImage, forKey: "inputImage")
                return filter.outputImage
        }
```

video process demo

- ImageLab++, filtering video in real time



updating filter parameters

- can be done on the fly, without performance loss

init

```
let filter:CIFilter = CIFilter(name: “CIBumpDistortion")
filter.setValue(-0.5, forKey: "inputScale")
filter.setValue(75, forKey: "inputRadius")
```

apply

```
self.visionAnalgesic.setProcessingBlock()
    {(inputImage:CIImage)->(CIImage) in
        self.filter.setValue(inputImage, forKey: "inputImage")
        return self.filter.outputImage
    }
```

updating filter parameters

- update from the UI

setup when users drags

get drag location

Transform from UI to CoreImage

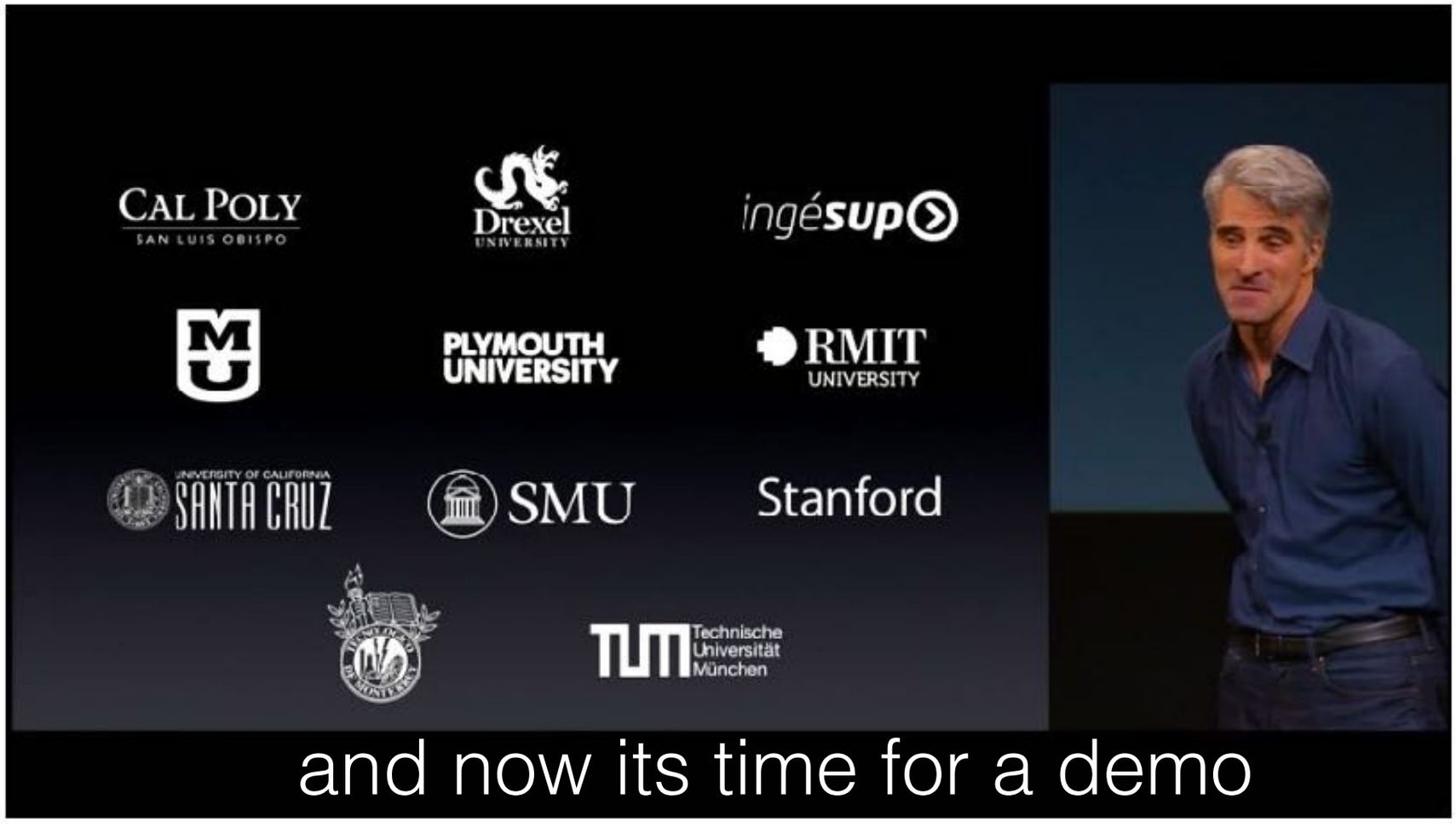
```
@IBAction func panRecognized(sender: UIPanGestureRecognizer) {
        let point = sender.locationInView(self.view)

      // this must be custom for each camera position and for each orientation
        let tmp = CIVector(x:point.y,y:self.view.bounds.size.width-point.x)

      filter.setValue(tmp, forKey: "inputCenter")
}
```

update center

transform differs for each combination of UI device position and camera position

filter param demo

- PinchMe



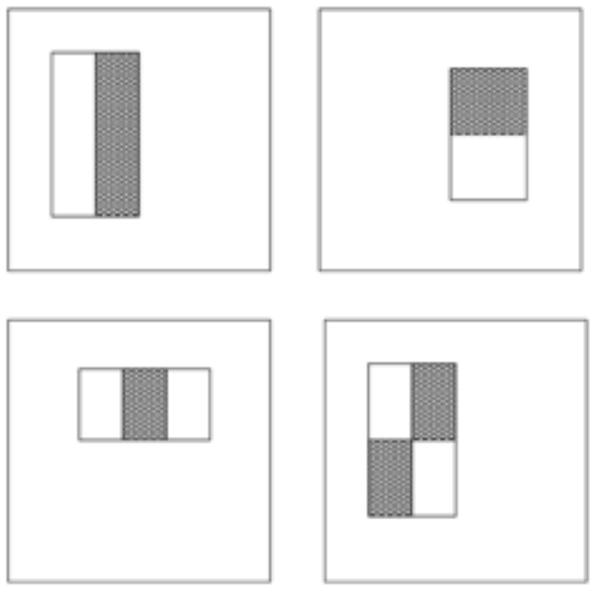
face detection

- is a face in the picture and where?
- algorithm prior to 2017: accelerated variant of Viola Jones
- after 2017: added a deep neural network
- essentially, a “matching” filter is applied
- only happens in one orientation
- but multiple scales (which takes “some” time)



an intuition: haar filters

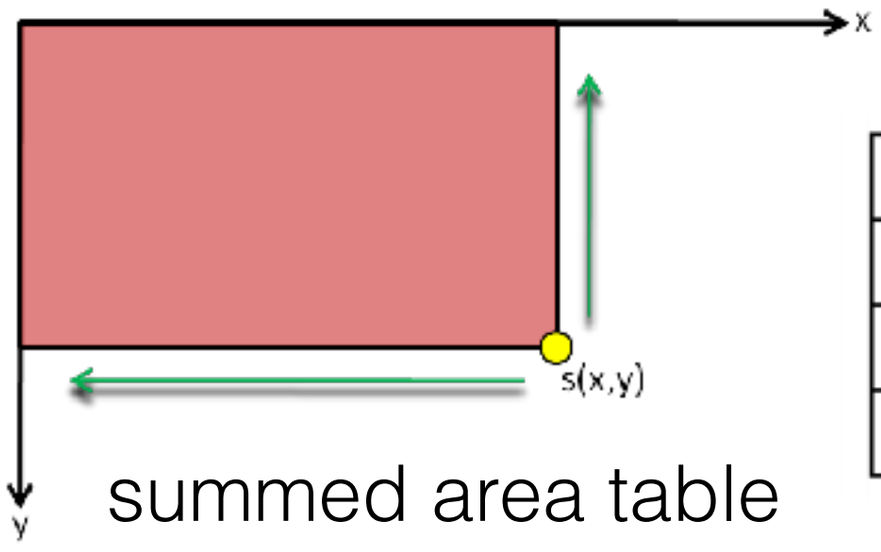
- face detection with “rectangle” features



feature value =
sum of pixels in white area -
sum of pixels in black area

“best” dark and light rectangles already chosen for face detection!

sum of any rectangle= C-D-B+A



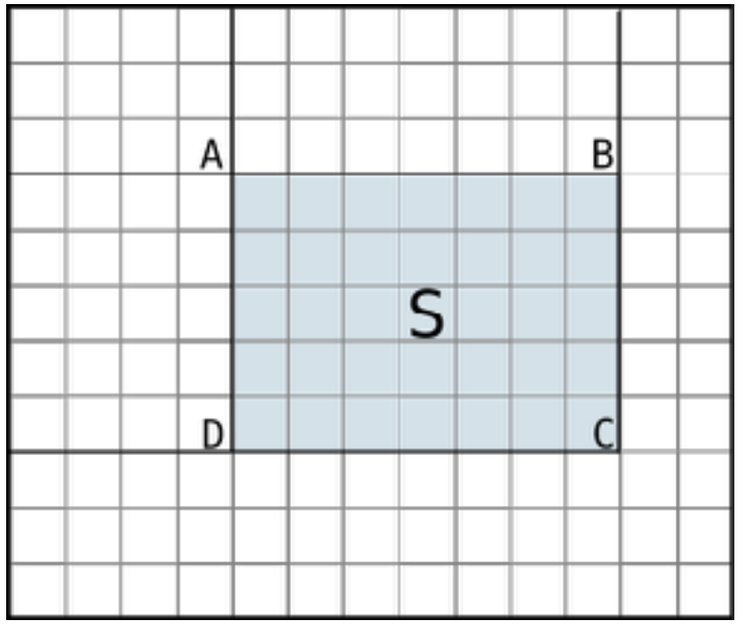
summed area table

4	1	2	2
0	4	1	3
3	1	0	4
2	1	3	2

original

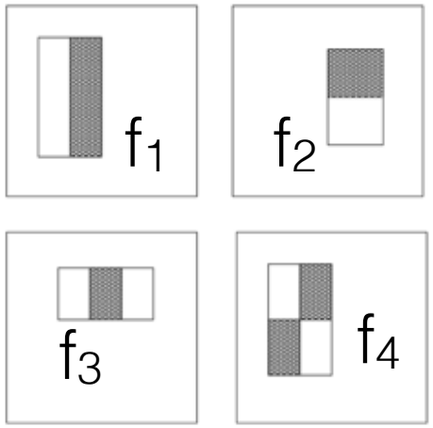
4	5	7	9
4	9	12	17
7	13	16	25
9	16	22	33

summed



the Viola Jones haar cascade

- Created in 2001: Viola, P. and Jones, M.J. Robust Real-time Object Detection Using a Boosted Cascade of Simple Features. CVPR 2001.
- train a bunch of “classifiers” with lots of examples



cascade these

$$\underbrace{C_t(x)}_{\text{classifier}} = \underbrace{1, \text{ if } f_t > \theta_t}_{\text{feature above thresh}}$$

combine output of classifiers

$$\underbrace{C(x)}_{\text{ensemble}} = \underbrace{1, \text{ if } \sum_{t=0}^{T-1} \alpha_t C_t(x) > \frac{1}{2} \sum_{t=0}^{T-1} \alpha_t}_{\text{learned weights}}$$

learning

- ML algorithms are tough to train
  - need examples in various lighting and illumination
  - different poses, glasses, with hair in face
  - different genders, races, and scales
  - **what made this easier?**
- easy to use once trained
  - just getting integral image
  - then getting relevant “features”
  - multiply with learned weights!
- iOS already has done the training for you

face detection iOS

- similar pipeline to applying a filter
- ...e CIDetector class
- specify wh...re the processing should occur

specify options

detector types: face, rectangle, QRCode, text

context

```
let optsDetector = [CIDetectorAccuracy:CIDetectorAccuracyHigh]

let detector = CIDetector(ofType: CIDetectorTypeFace,
                  context: self.videoAnalgesic.getCIContext(),
                  options: optsDetector)

var optsFace = [CIDetectorImageOrientation:self.visionAnalgesic.ciOrientation)]
```

orientation

for each face

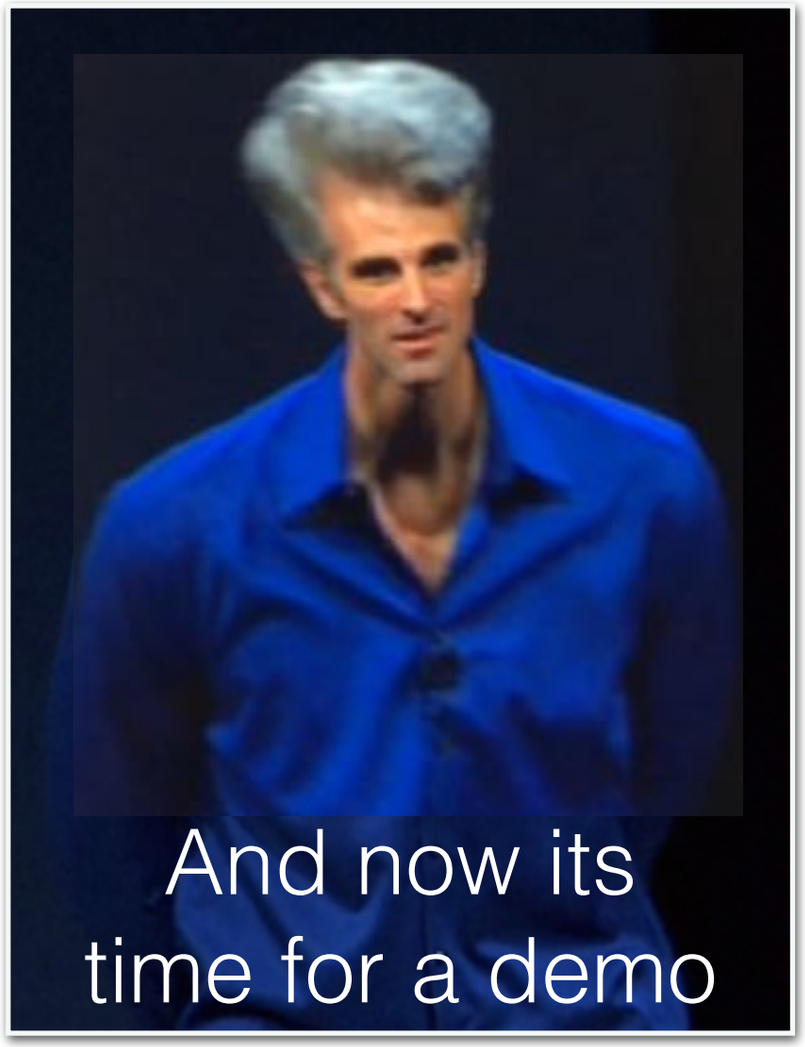
```
var features = detector.featuresInImage(inputImage, options: optsFace)

for f in features as [CIFaceFeature]{
      NSLog("%@",f)
}
```

do this

options specific to "run"

face demonstration

- PinchMe++

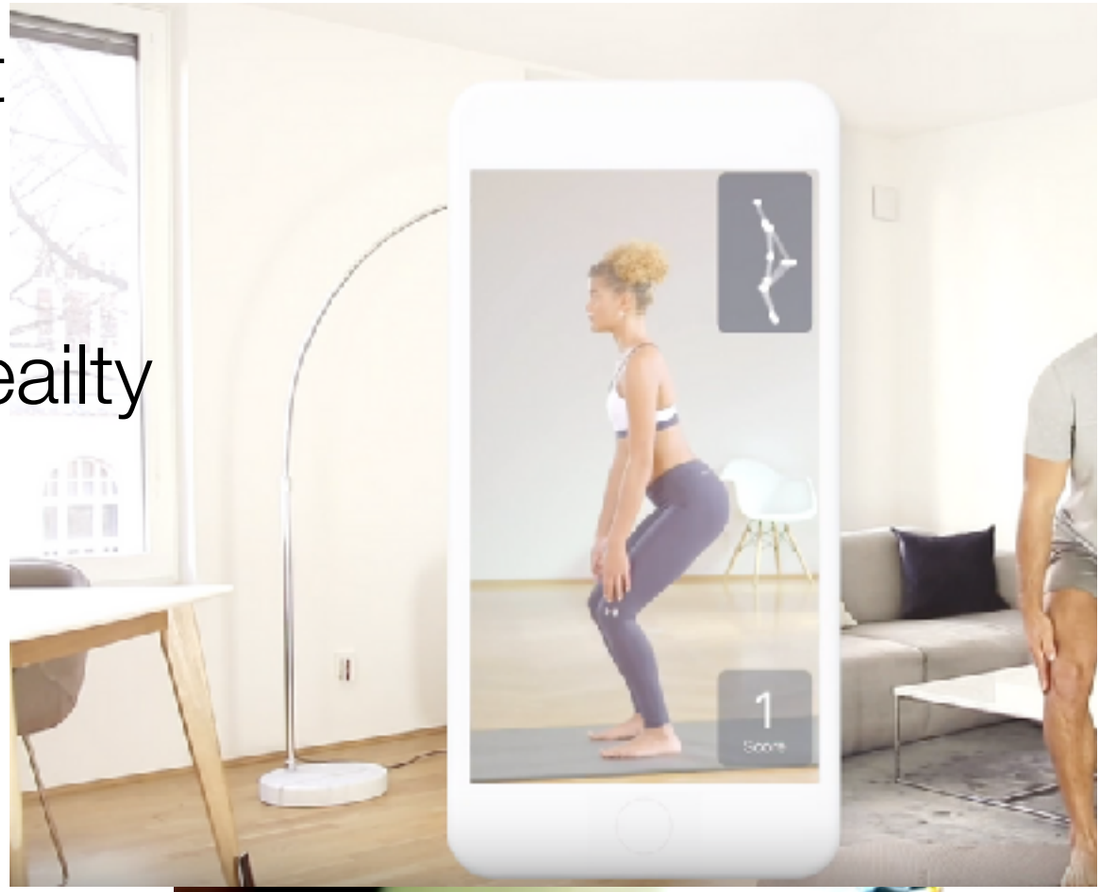


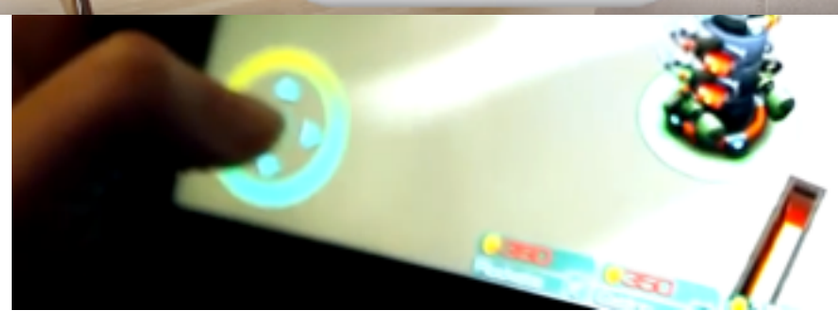
computer vision

- face detection is just the beginning
  - could use tracking method for any object
- could also do “recognition”
  - typically done with eigen-faces or fisher-faces
    - would take (slightly) too much time in this class to implement
- more than just tracking
  - edge detection
  - finding lines and shapes
  - color space transformations
- extract “knowledge” from a scene

computer vision in iOS

- mobile camera is a rich medium for:
  - enhancement
  - interaction
  - augmented reailty
  - gaming
  - tracking
- health



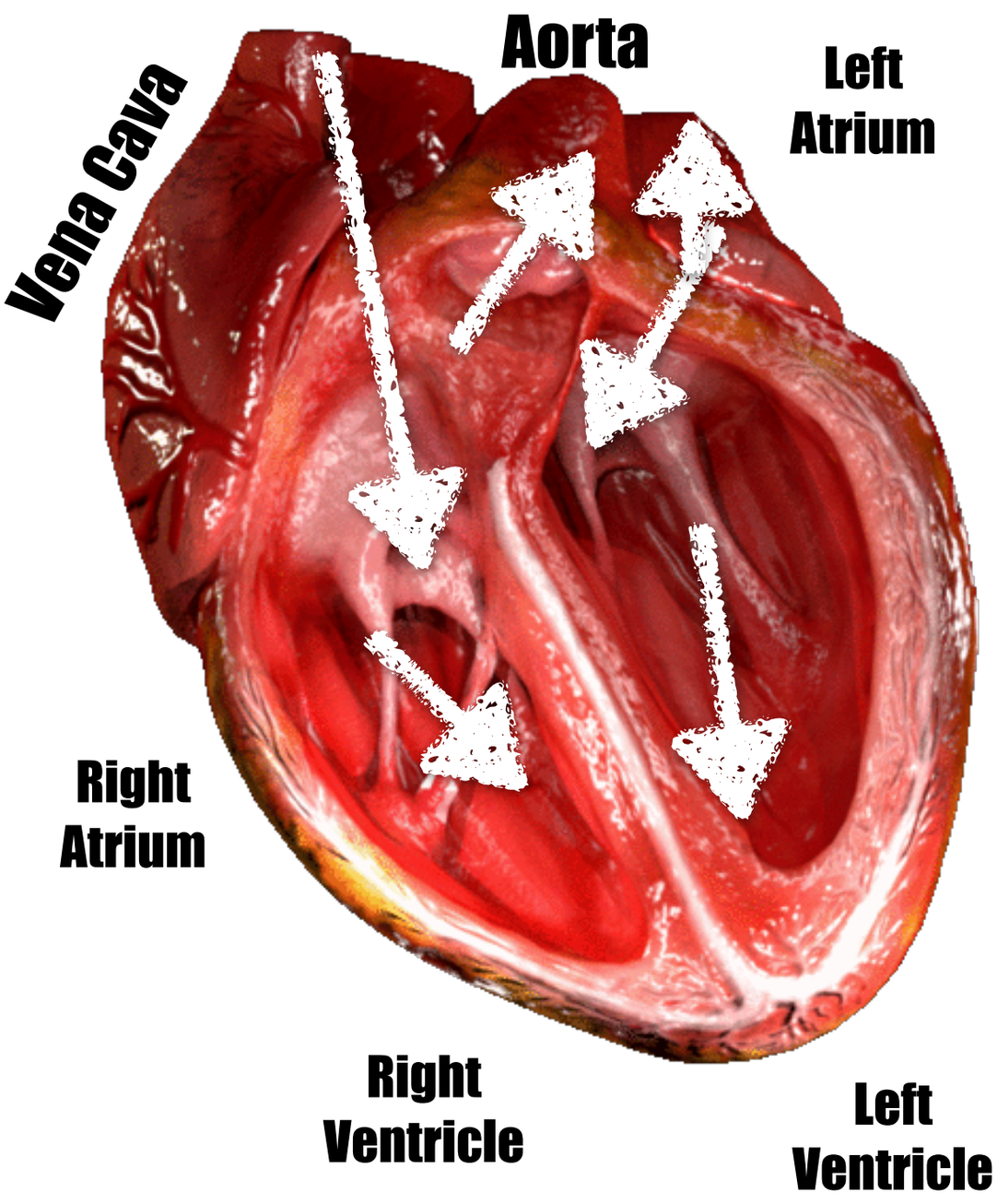


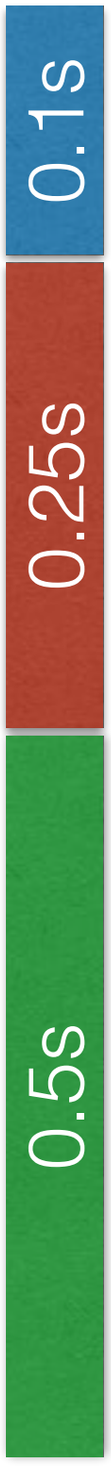
health?

- detecting heart rate from the camera
- what is the function of the heart?
  - pump oxygenated blood from lungs to the rest of the body
  - bring back de-oxygenated blood
- a pump maintains pressure and flow
  - no pump works continuously
  - series of pressure buildup, release, buildup, release
  - cycles in the heart is the **heart beat**

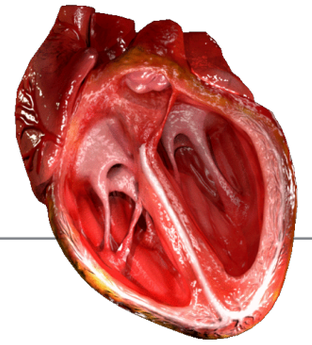
Image Credit: Wikipedia

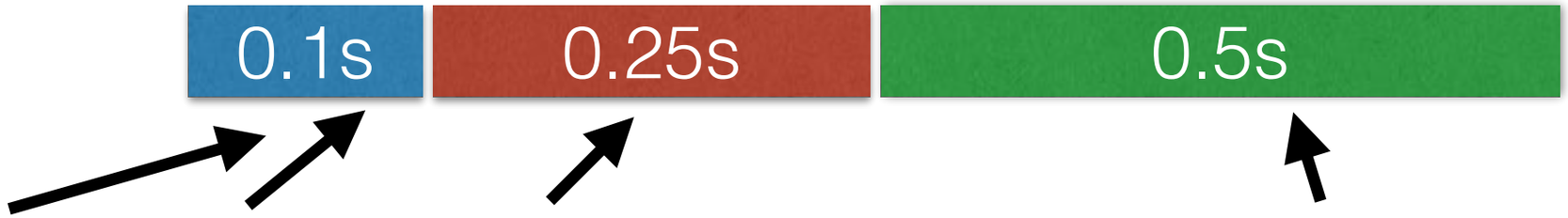
the cardiac cycle



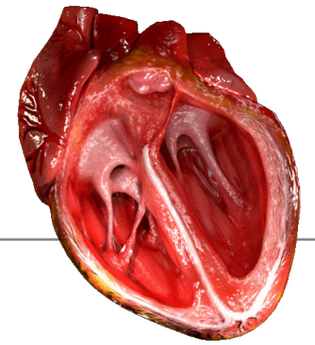


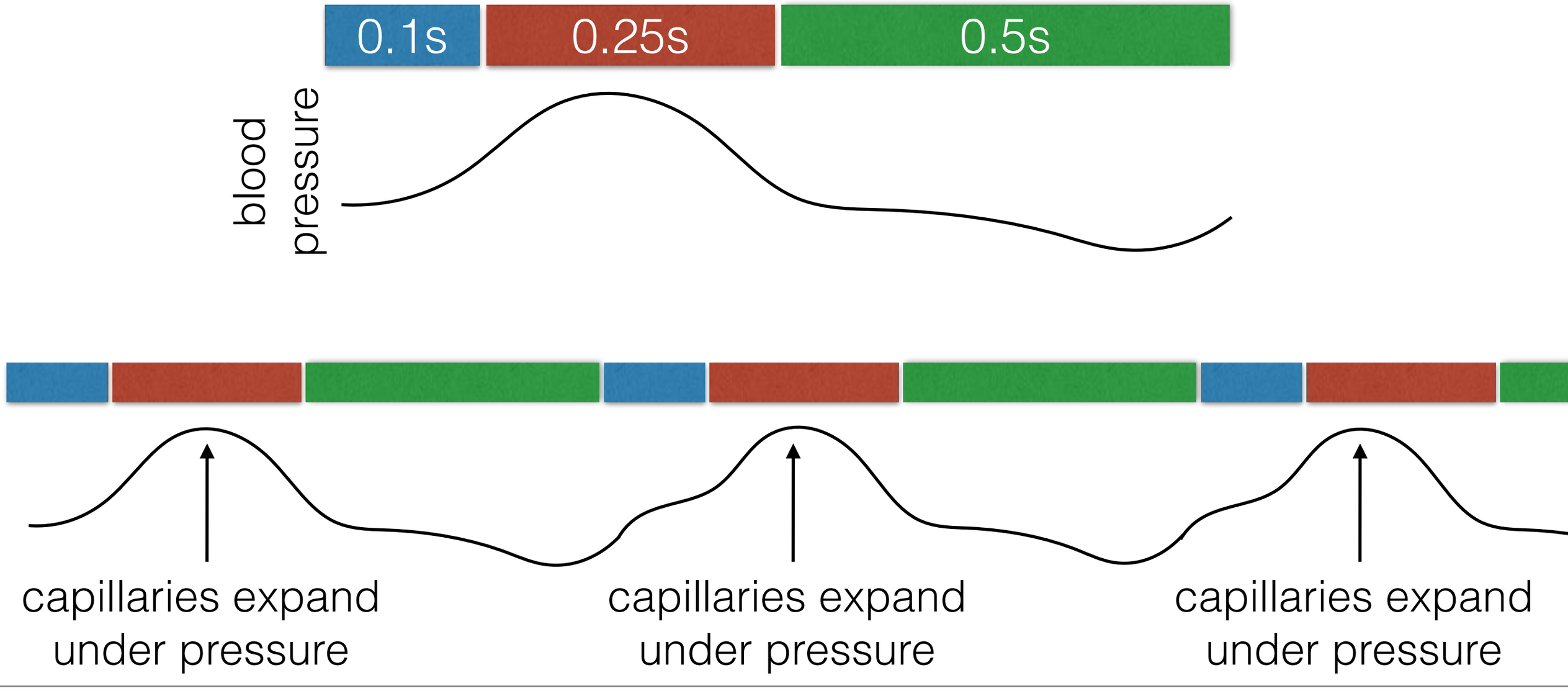
- aortic valve and vena cava valve open
- blood enters ventricles
- pump of heart ejects blood into arteries and out to lungs/body
- heart relaxes, letting blood flow into atria and ventricles



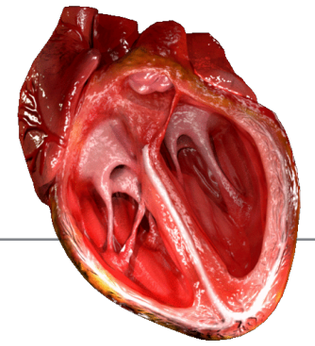

0.1s
0.25s
0.5s

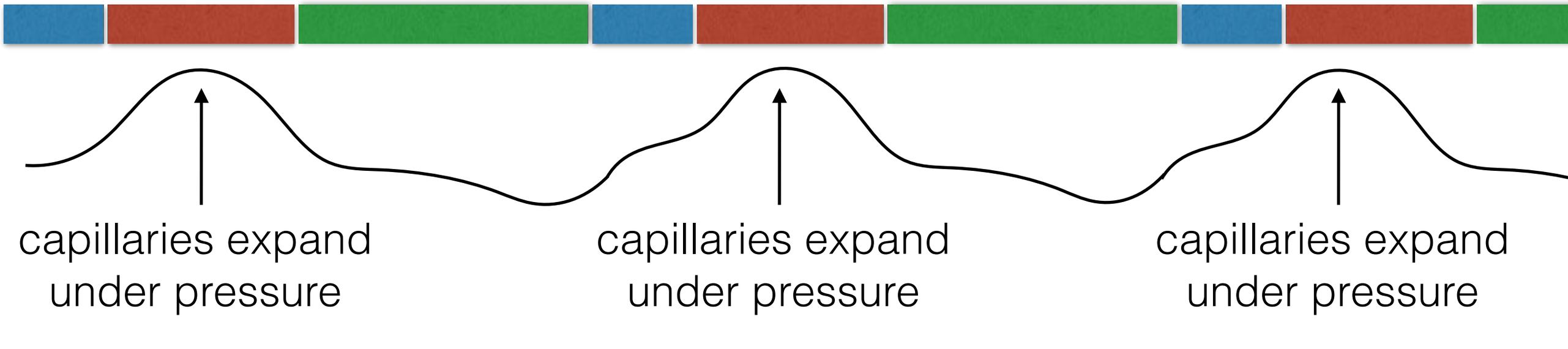
a signal from the heart





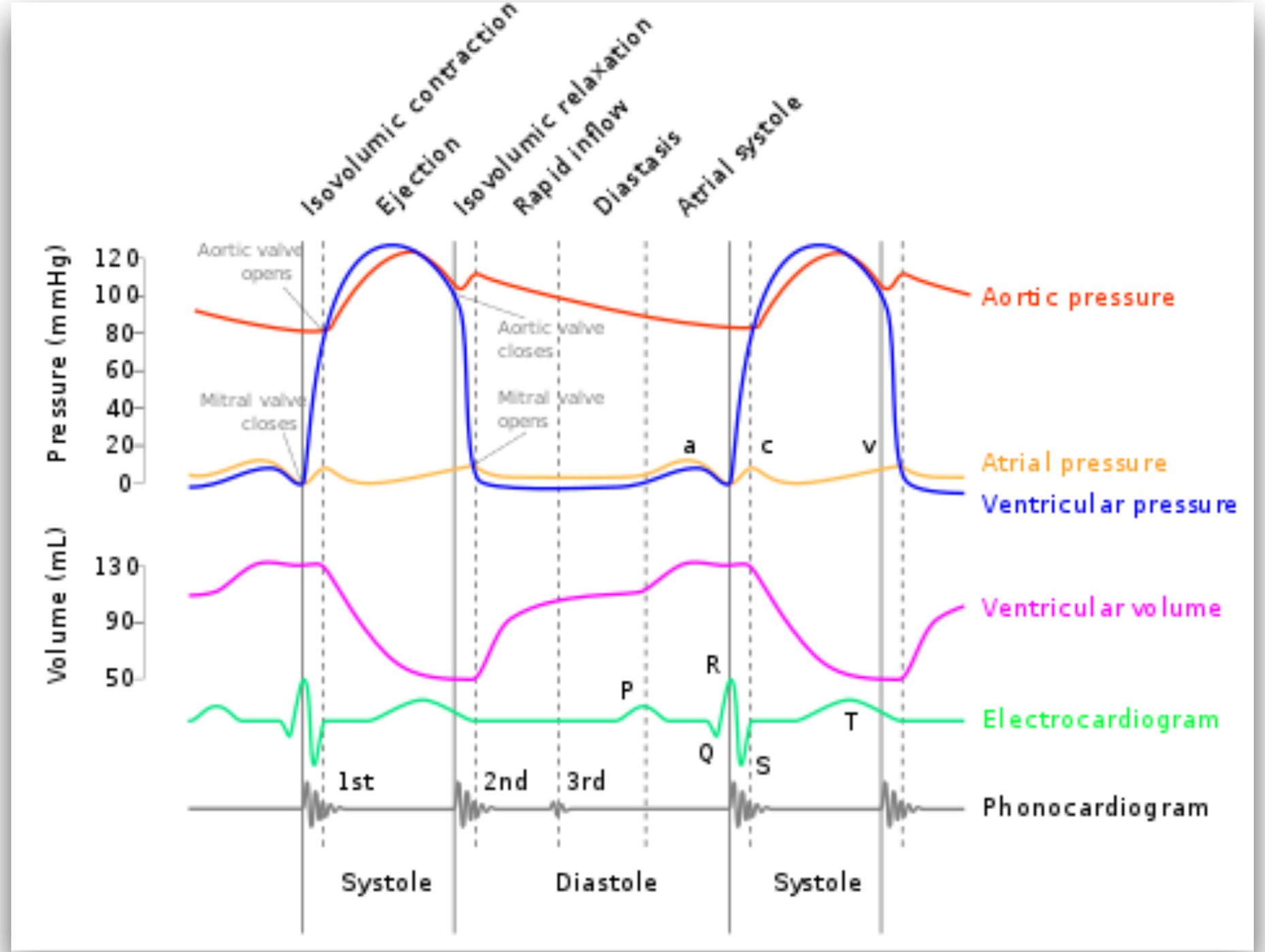
a signal from the heart





- capillary expansion means more blood under skin
  - shift in redness from oxygenated blood
  - shift in blueness from deoxygenated blood
  - more blood molecules for light to reflect from

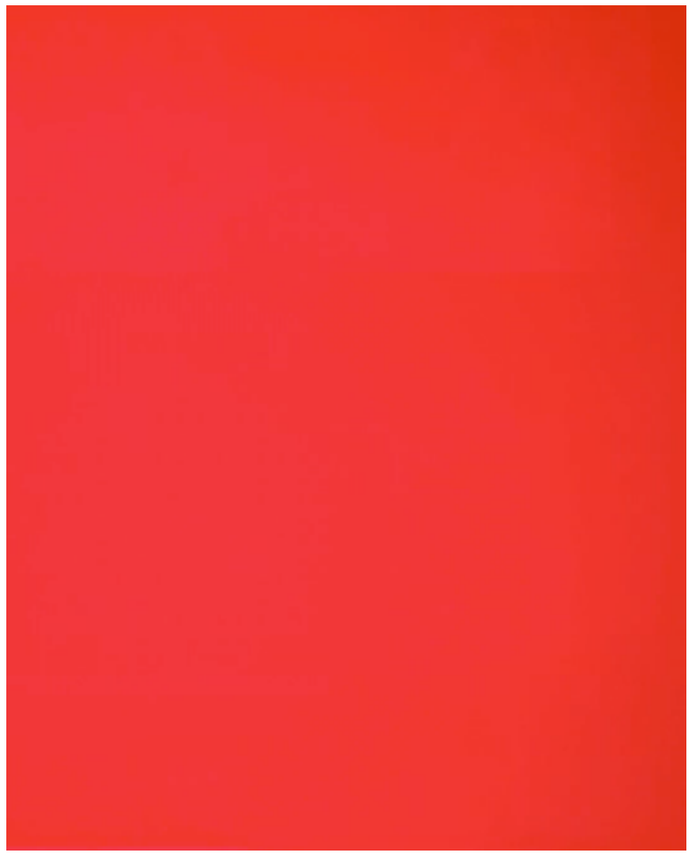
many signals…



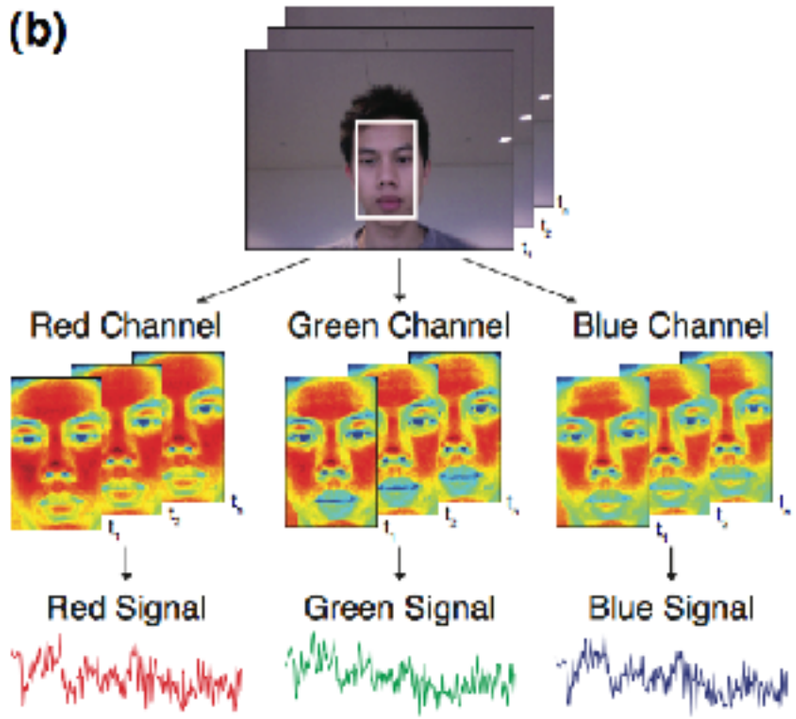
https://en.wikipedia.org/wiki/Cardiac_cycle

a signal from the heart

- hold finger over
  - camera
  - torch (always on flash)



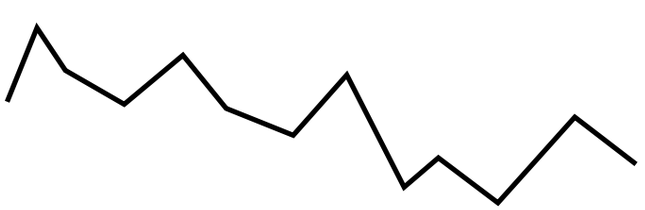
exemplary work



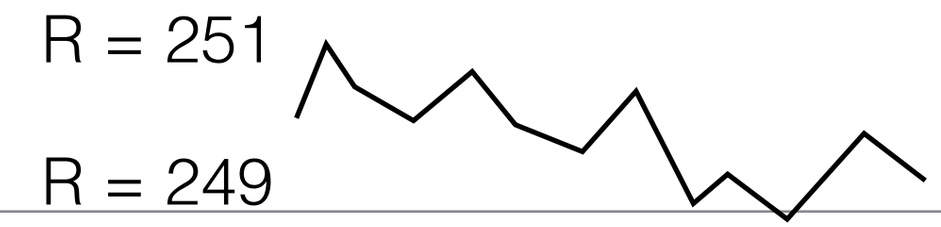
photoplethysmography (PPG)

R = 251

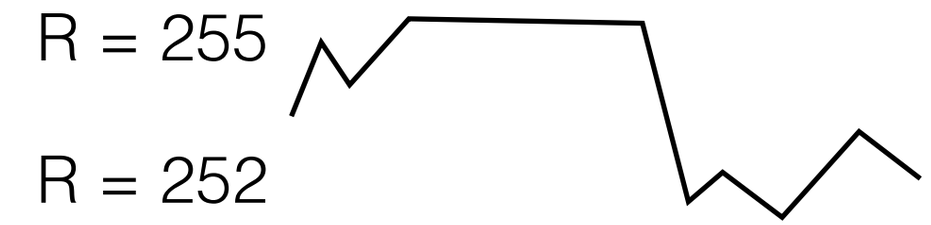
R = 249



caveats



- do not press too hard on camera or make flash bright
- vasoconstriction and vasodilation
- bigger surface areas are better
- don’t move around too much



- the heart is not the only organ that increases pressure
  - what else could cause the capillaries to expand/contract?

- what method might you use to measure PPG from the camera?

a cool example

September 7, 2016

HemaApp screens for anemia, blood conditions without needle sticks

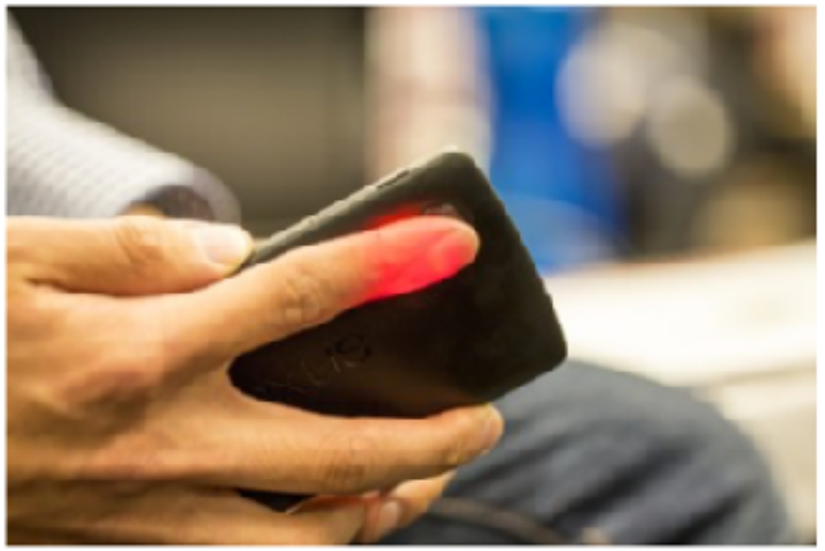
Jennifer Langston

News and Information

In the developing world, anemia — a blood condition exacerbated by malnutrition or parasitic disease — is a staggeringly common health problem that often goes undiagnosed.

In hospitals everywhere, children and adults with leukemia and other disorders require frequent blood draws to determine if they need blood transfusions.

In both cases, doctors are interested in measuring hemoglobin, a protein found in



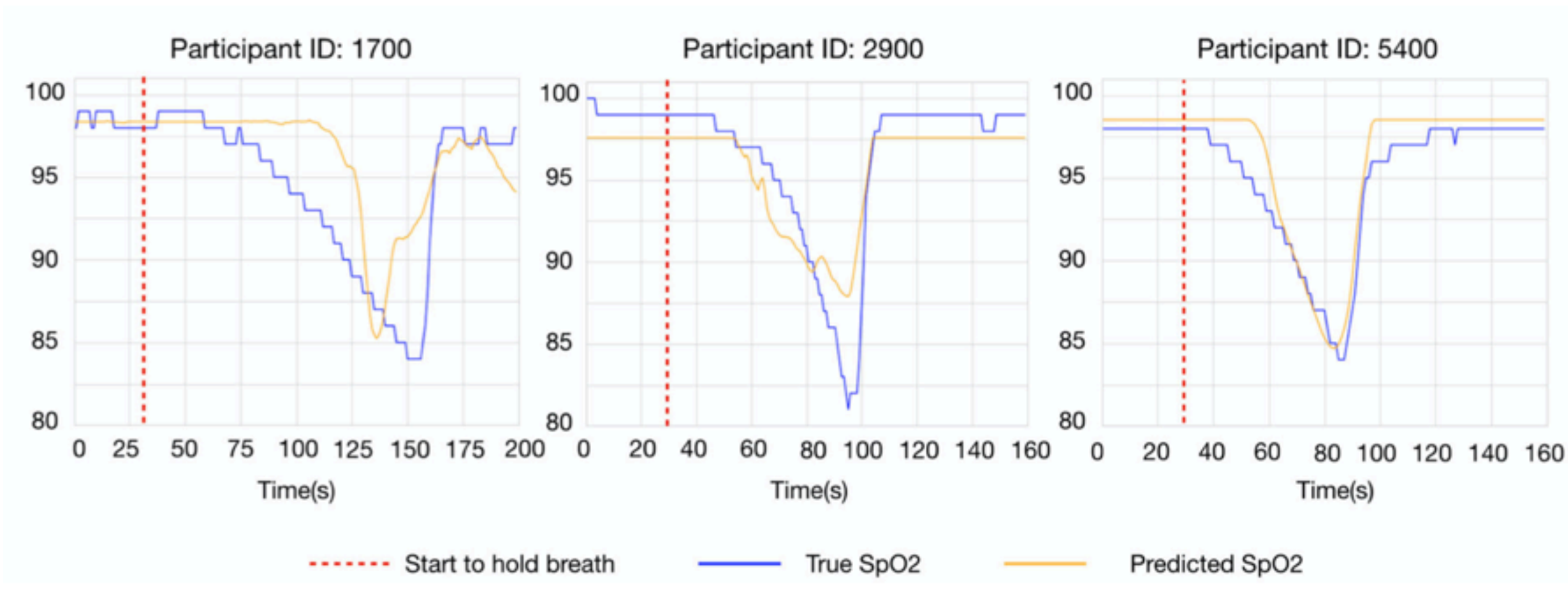
HemaApp measures hemoglobin levels and screens for anemia non-invasively by illuminating the patient's finger with a smartphone's camera flash. ***Dennis Wise/University of Washington***

a research project

Measuring Oxygen Saturation With Smartphone Cameras Using Convolutional Neural Networks

Xinyi Ding, Damoun Nassehi, and Eric C. Larson

Fig. 1. Experimental setup of ground truth pulse oximetry data collection and custom phone application.



follow on work…

ARTICLE OPEN

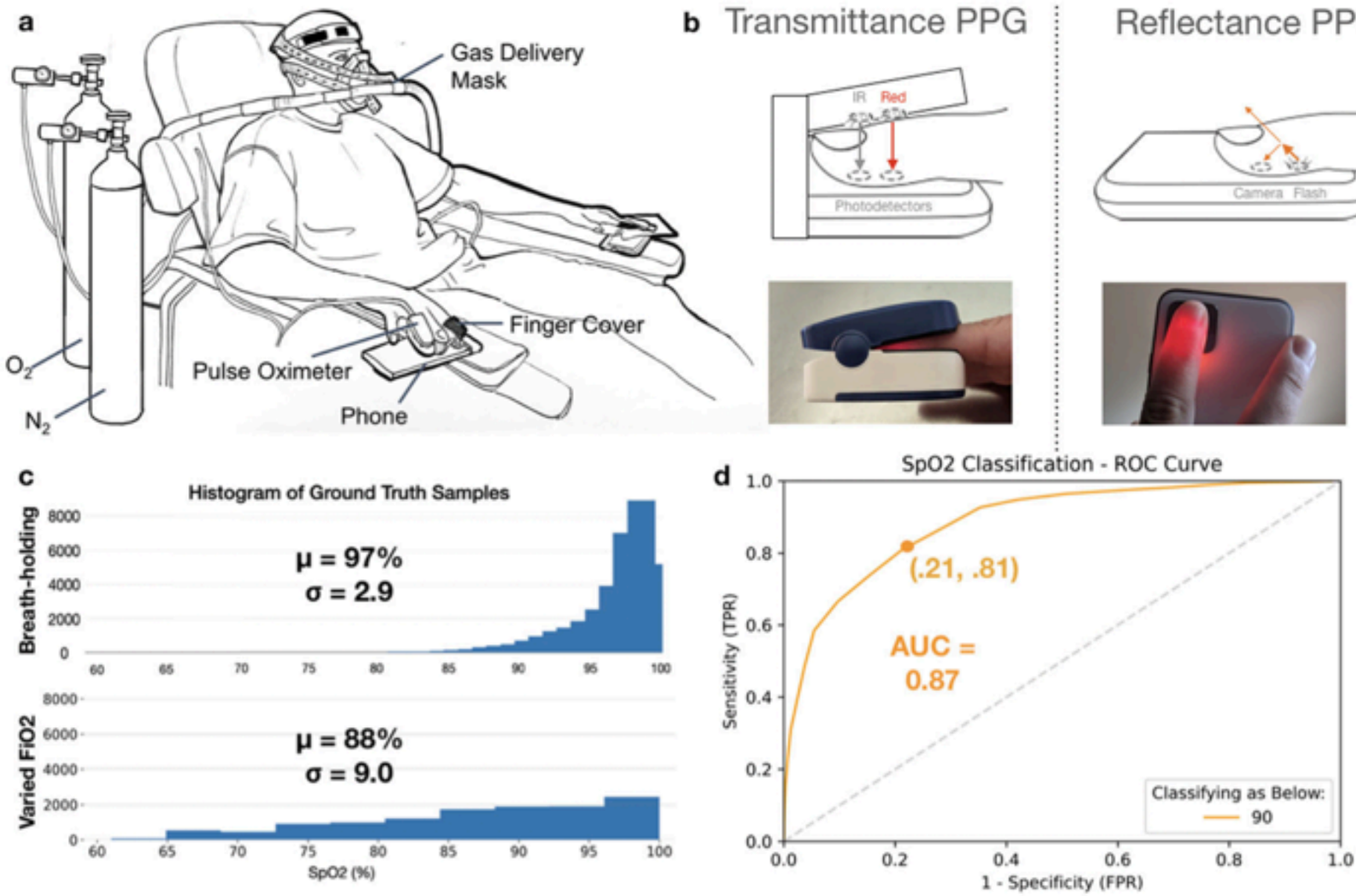
Check for updates

Smartphone camera oximetry in an induced hypoxemia study

Jason S. Hoffman[1,7]✉, Varun K. Viswanath[2,3,7], Caiwei Tian[1], Xinyi Ding[4], Matthew J. Thompson[5], Eric C. Larson[4], Shwetak N. Patel[1,6] and Edward J. Wang[2,3]

Hypoxemia, a medical conditior
leading indicator for dangerous
oximeters can provide accurate
capability in unmodified smartp
their health. Towards this goal,
sensing system using a varied f
smartphone-based contact PPG
learning model using this data
with 81% sensitivity and 79% s

npj Digital Medicine (2022)5:146

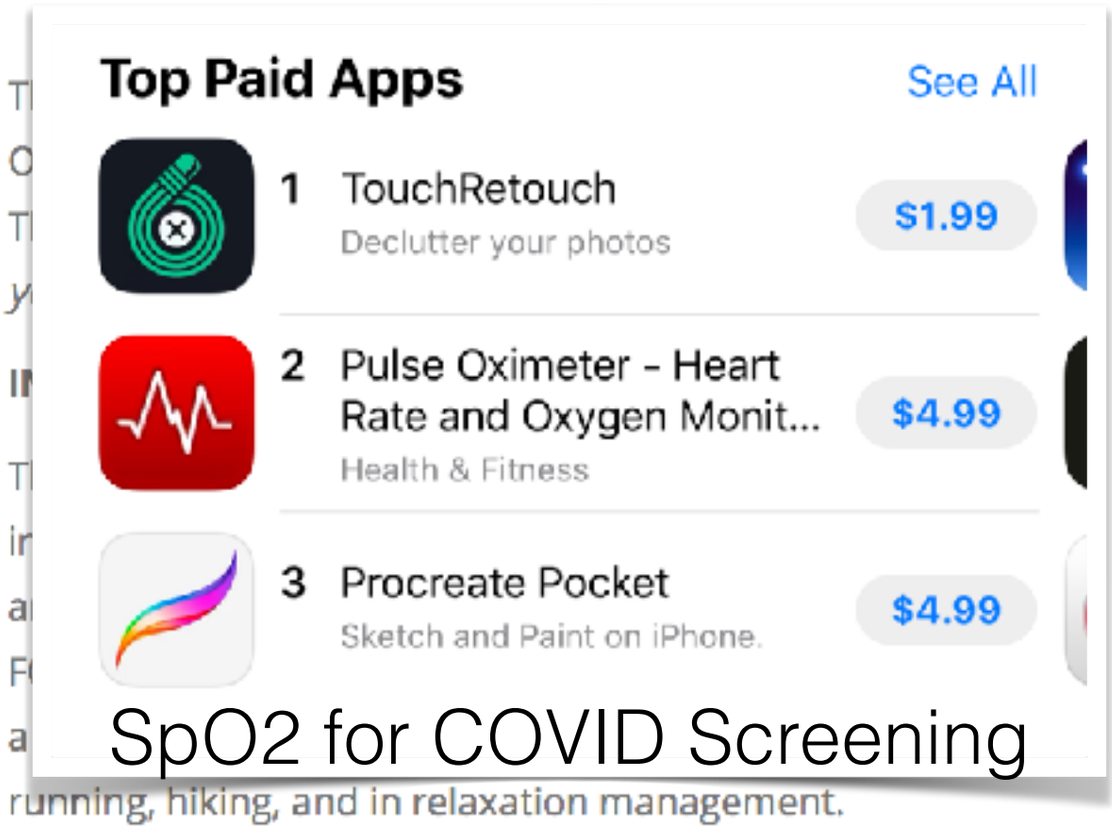


a short lived example

digiDoc Technologies

Home Products

The World's Only Digital Solution



SpO2 for COVID Screening

running, hiking, and in relaxation management.

Pulse Oximeter uses your iPhone's camera to detect your pulse and oxygen levels from your fingertip. Track and record heartbeat and blood oxygen levels. Instant results, easy to use, simple charts to save your progress.

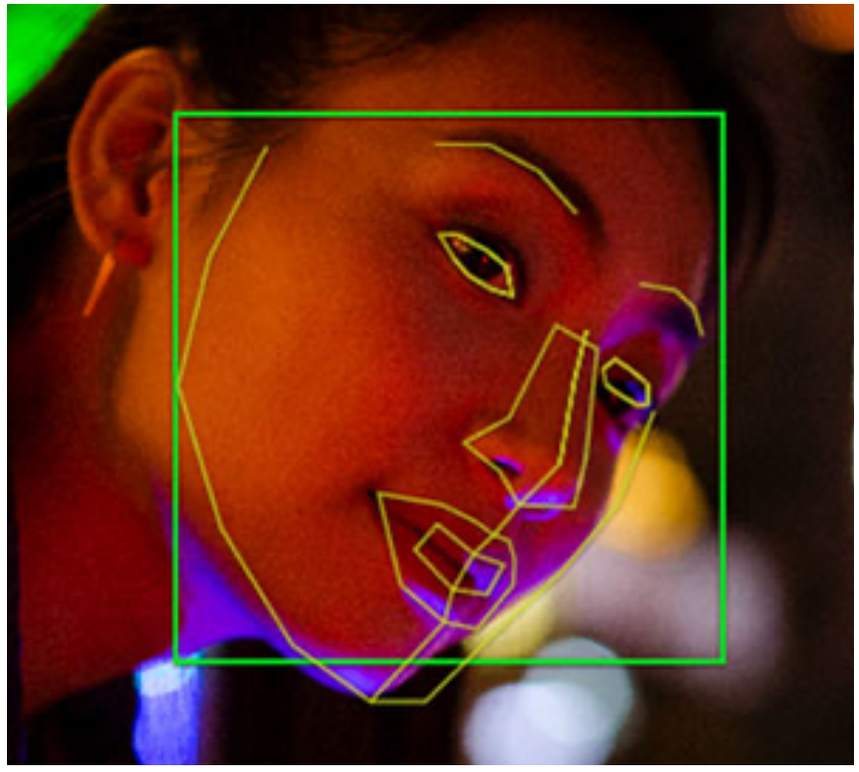
Features

- Measure pulse and blood oxygen saturation
- Pulse Oximeter range 93-100%
- Record and store history of data
- Real-time PPG graph for immediate accuracy
- Apple HealthKit integration
- Label selection

Download on the App Store

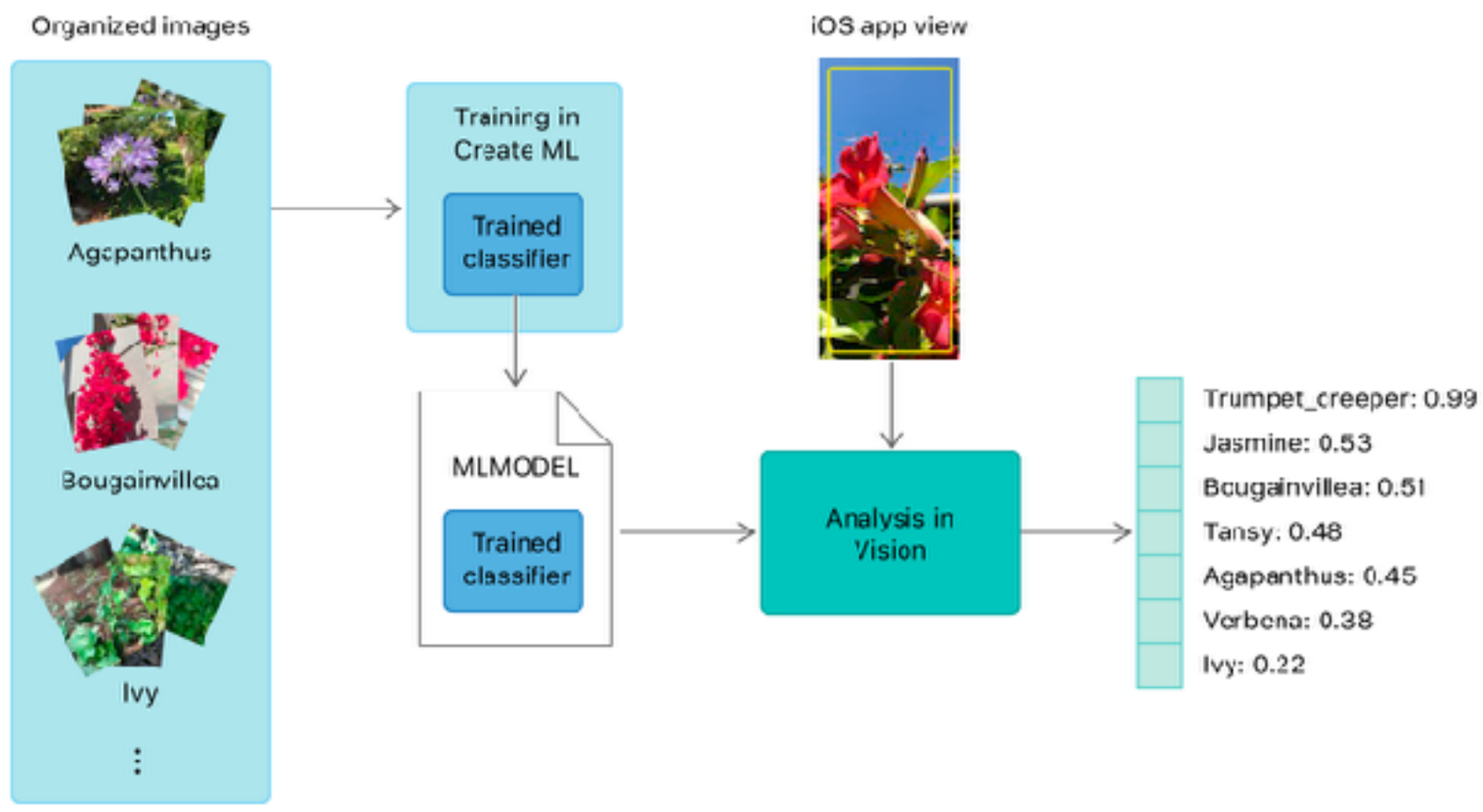
before moving on, recall:

- next class is the flipped module!
- computer vision with AppleVision
  - watch video lecture
  - come ready to add functionality to app next time



Classification with vision API

- load ml model in Xcode
- wrap model
- create vision request
- wait for result in completion handler



the vision API

```
// generate request for vision and ML model
let request = VNCoreMLRequest(model: self.model,
                              completionHandler: resultsMethod)

// add data to vision request handler
let handler = VNImageRequestHandler(cgImage: cgImage!, options: [:])

// now perform classification
do{
     try handler.perform([request])
}catch _{
     …
}

func resultsMethod(request: VNRequest, error: Error?) {
        guard let results = request.results as? [VNClassificationObservation]
            else { fatalError() }

        for result in results {
            if(result.confidence > 0.05){
                print(result.identifier,result.confidence)
            }
        }
    }
```

setup vision request with completion handler and ML model

add data to request(s)

perform request

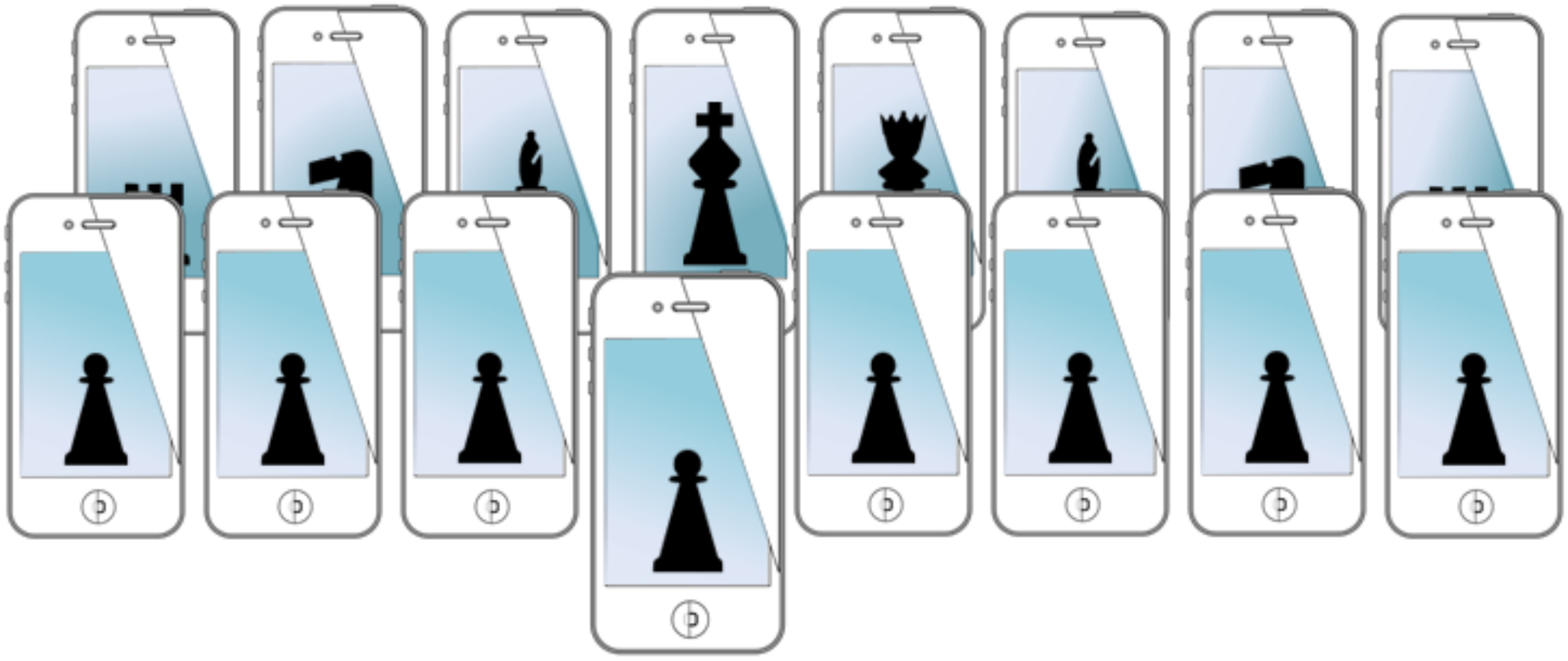
interpret request results

CoreML, a taste (if time)

- demo using vision API (if time)

https://github.com/SMU-MSLC/CoreMLVision

MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

computer vision with core image

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