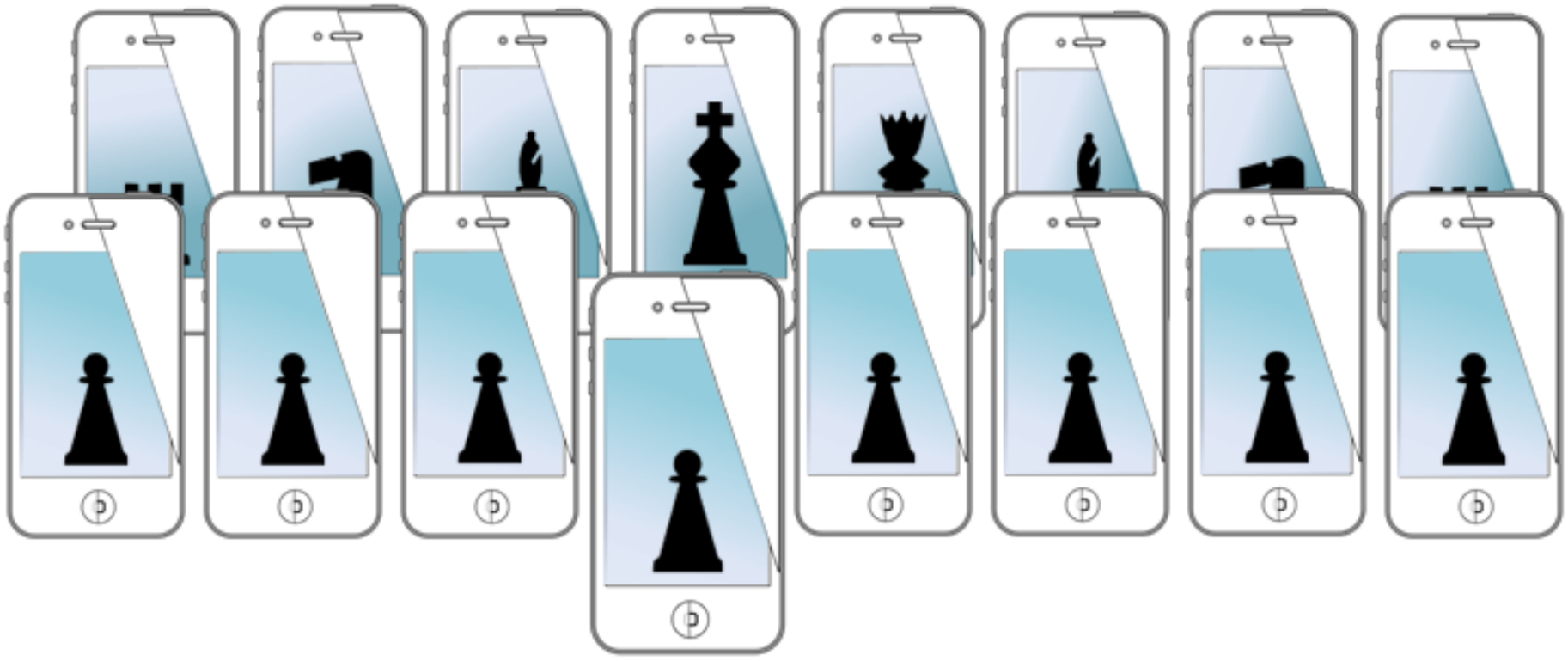
MOBILE SENSING LEARNING



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:
  - grades are coming, A3 is due soon!
- Agenda:
  - video processing
  - computer vision
    - face detection
    - heart physiology

how to program this?

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

declare

```
var videoAnalgesic:VideoAnalgesic! = nil
```

init

```
self.videoAnalgesic = VideoAnalgesic(mainView:self.view)
```

start

```
if !self.videoAnalgesic.isRunning{
    self.videoAnalgesic.start()
}
```

options

```
self.videoAnalgesic.setPreset(AVCaptureSessionPreset.medium)
self.videoAnalgesic.toggleCameraPosition()
self.videoAnalgesic.setCameraPosition(AVCaptureDevicePosition.front)
```

stop

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

VideoAnalgesic

- 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.videoAnalgesic.setProcessingBlock()
             {(inputImage:CIImage)->(CIImage) in
                 return inputImage
         }
```

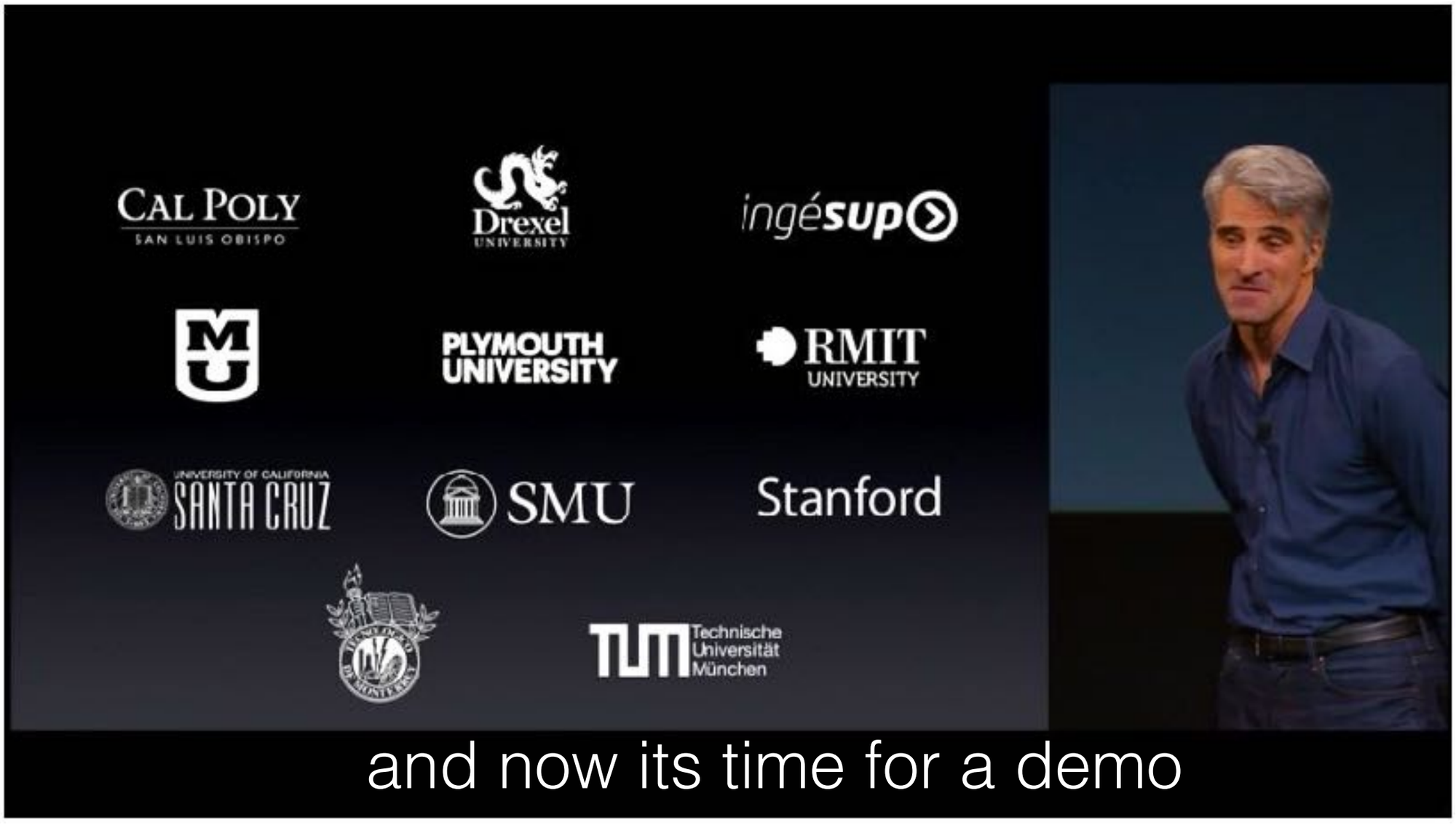
image from camera passed in

return image to draw to screen

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

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

video process demo

- ImageLab++



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.videoAnalgesic.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

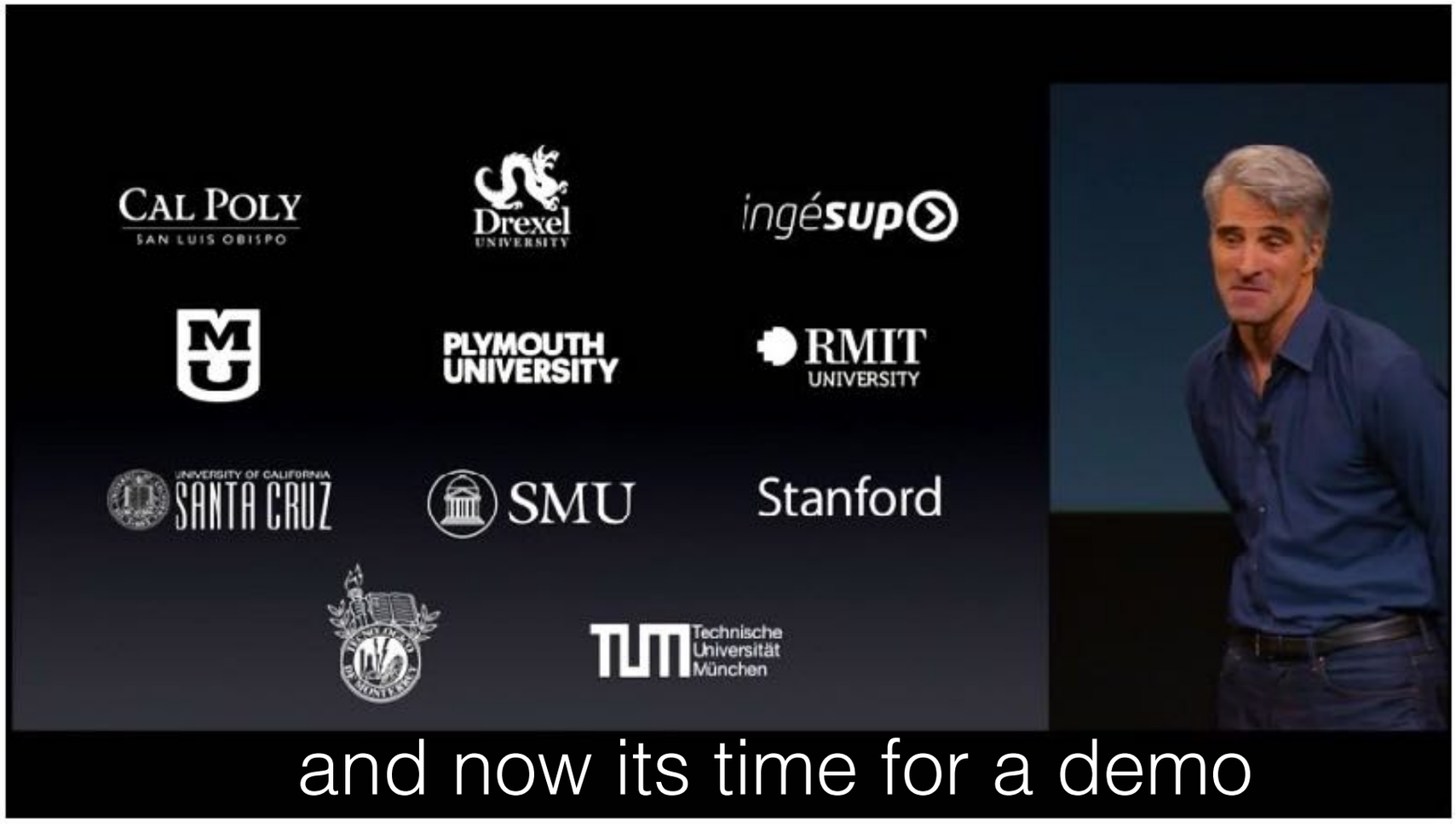
```
@IBAction func panRecognized(sender: UIPanGestureRecognizer) {
        let point = sender.locationInView(self.view)

      // 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

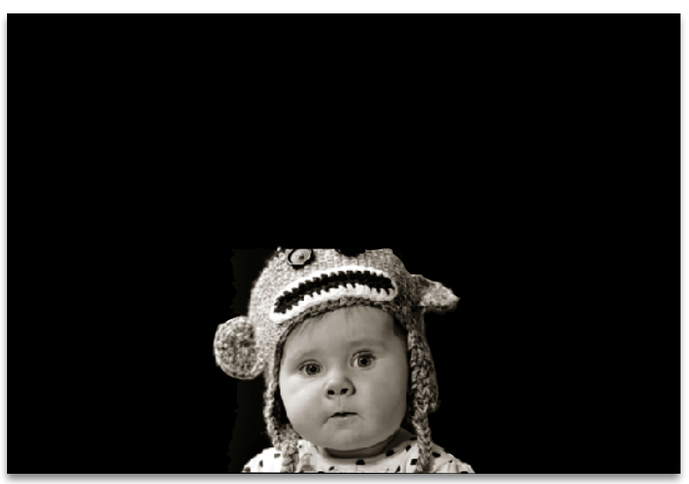
- PinchMe



face detection

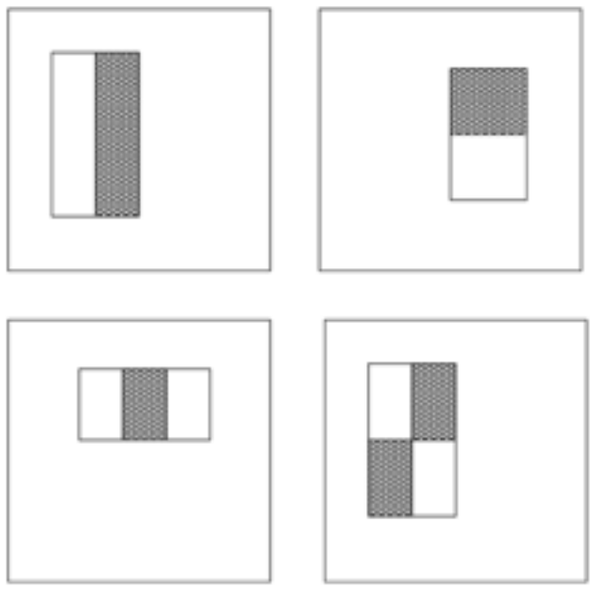
- is a face in the picture and where?
- algorithm prior to 2017: accelerated variant of Viola Jones
- after 2017: deep neural network
- essentially, a “matching” filter is applied
- only happens in one orientation
- but multiple scales (which takes “some” time)





an intuition: haar filters

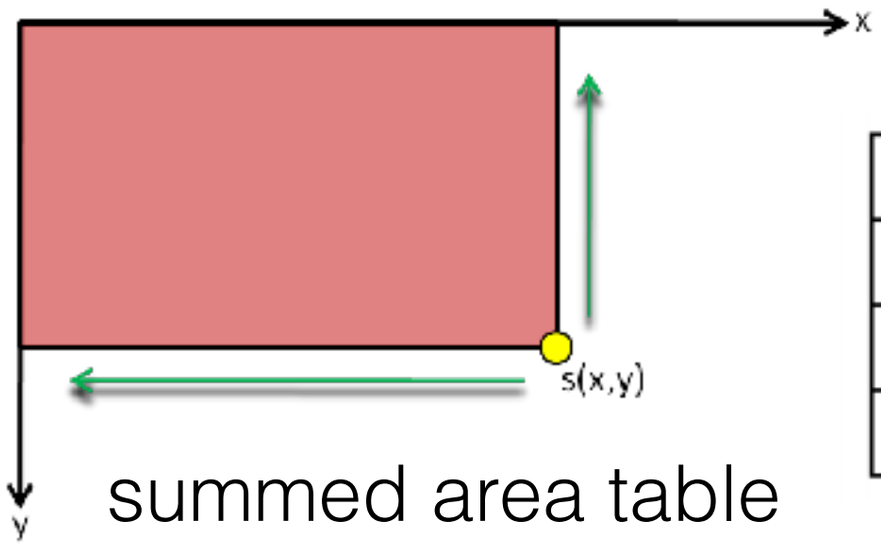
- face detection with “rectangle” features



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



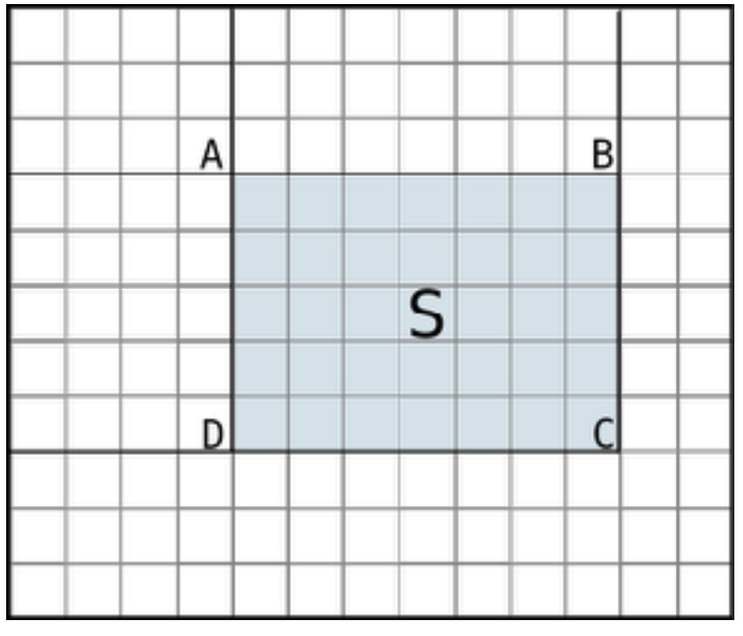
summed area table

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

original

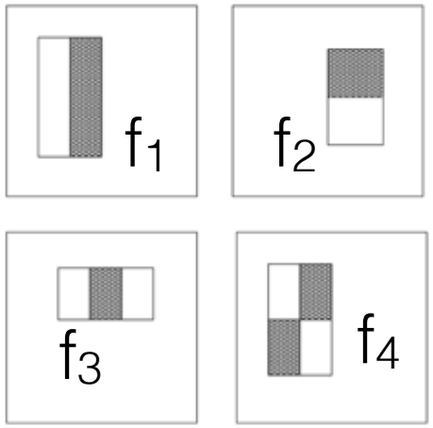
4	5	7	9
4	9	12	17
7	13	16	25
9	16	22	33

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

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

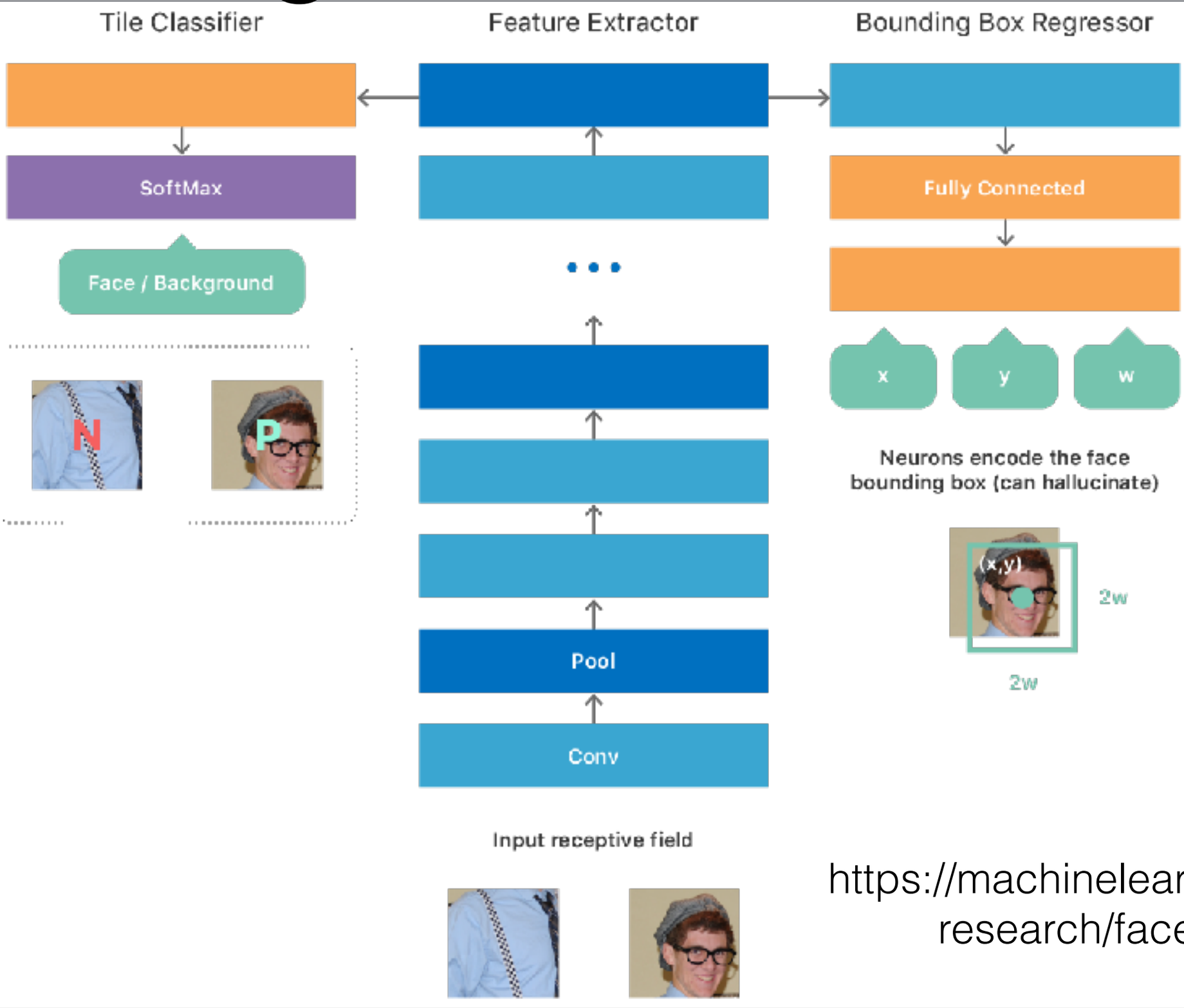
combine output of classifiers

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

learning

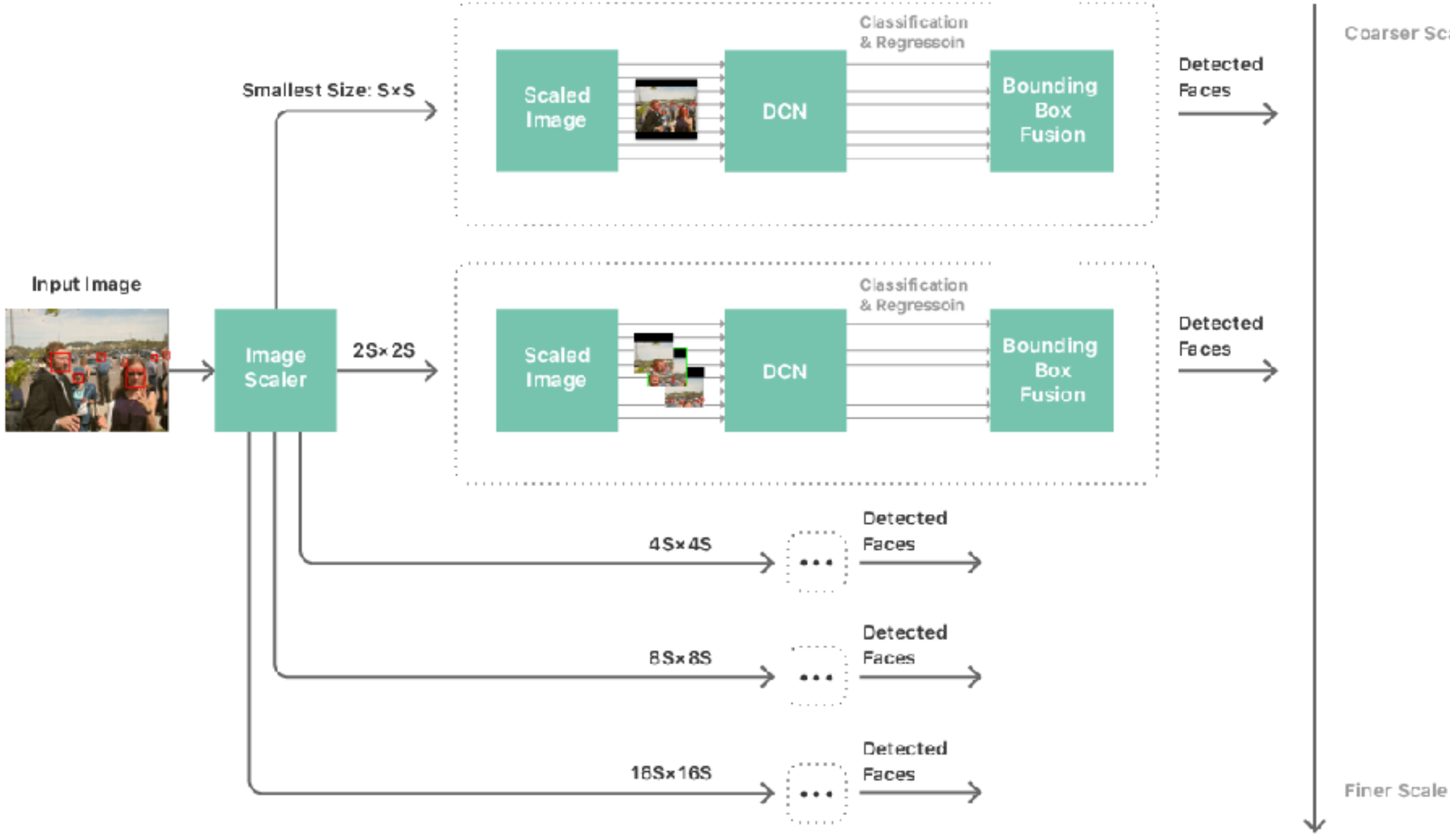
- ML algorithms are tough to train
  - need examples in various lighting and illumination
  - different poses, glasses, with hair in face
  - different genders, races, and scales
  - **what made this easier?**
- easy to use once trained
  - just getting integral image
  - then getting relevant “features”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

moving on from Viola Jones



https://machinelearning.apple.com/research/face-detection

moving on from Viola Jones



https://machinelearning.apple.com/research/face-detection

face detection iOS

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

specify options

detector type: face, rectangle, QRCode, Text

context

```
let optsDetector = [CIDetectorAccuracy:CIDetectorAccuracyHigh]

let detector = CIDetector(ofType: CIDetectorTypeFace,
                  context: self.videoAnalgesic.getCIContext(),
                  options: optsDetector)
var optsFace =
[CIDetectorImageOrientation:self.videoAnalgesic.getImageOrientationFromUIOrientation(UIApp
lication.sharedApplication().statusBarOrientation)]
```

orientation

for each face

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

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

options specific to "run"

do this

face demonstration

- PinchMe++