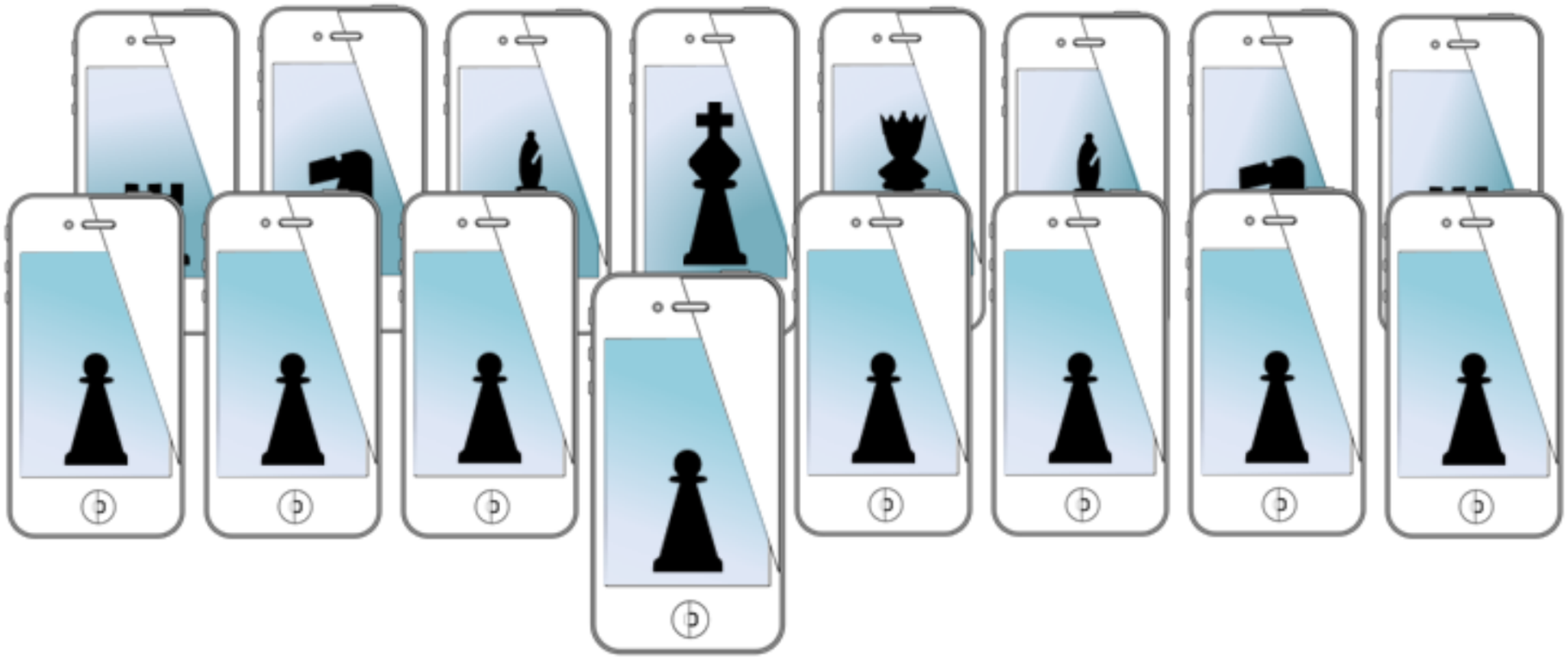
MOBILE SENSING LEARNING



CSE5323 & 7323

Mobile Sensing and Learning

week 6: core image and image processing

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

course logistics/agenda

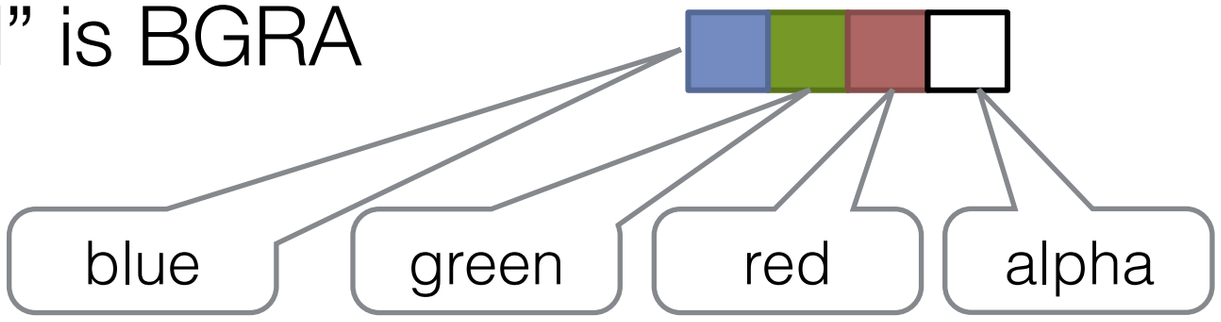
- logistics:
  - A2 grades coming
  - A3 will be due Friday
- today:
  - image processing basics
  - core image filtering

what is image processing

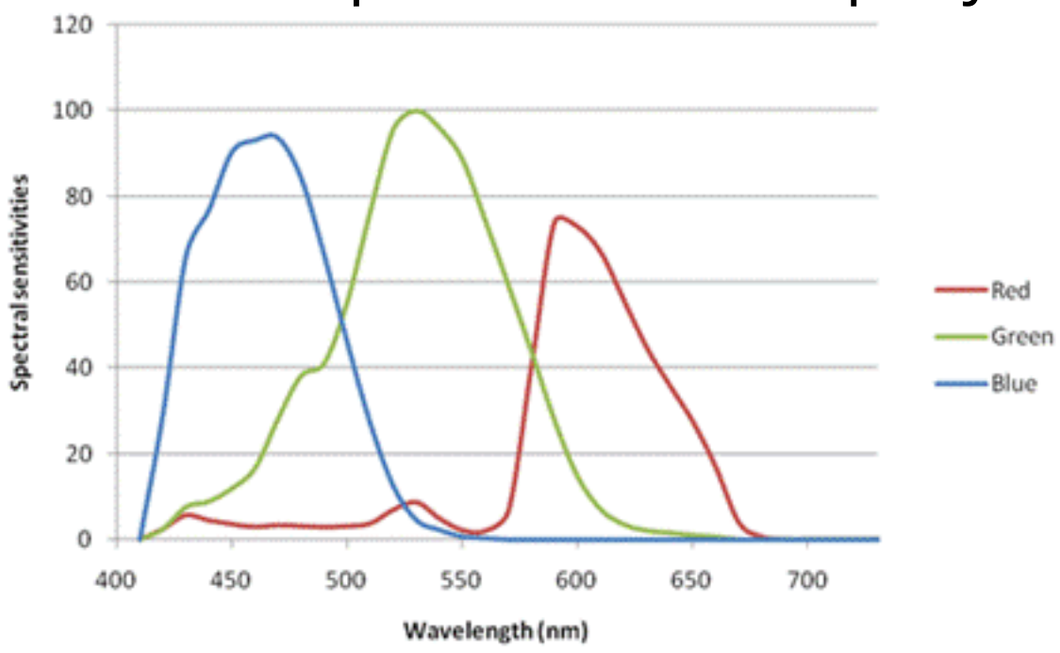
- the **art** and **science** of manipulating pixels
  - combining images (blending or compositing)
  - enhancing edges and lines
  - adjusting contrast, color
  - warping, transformation
  - filtering
  - …anything you can do in photoshop
  - also used in computer vision

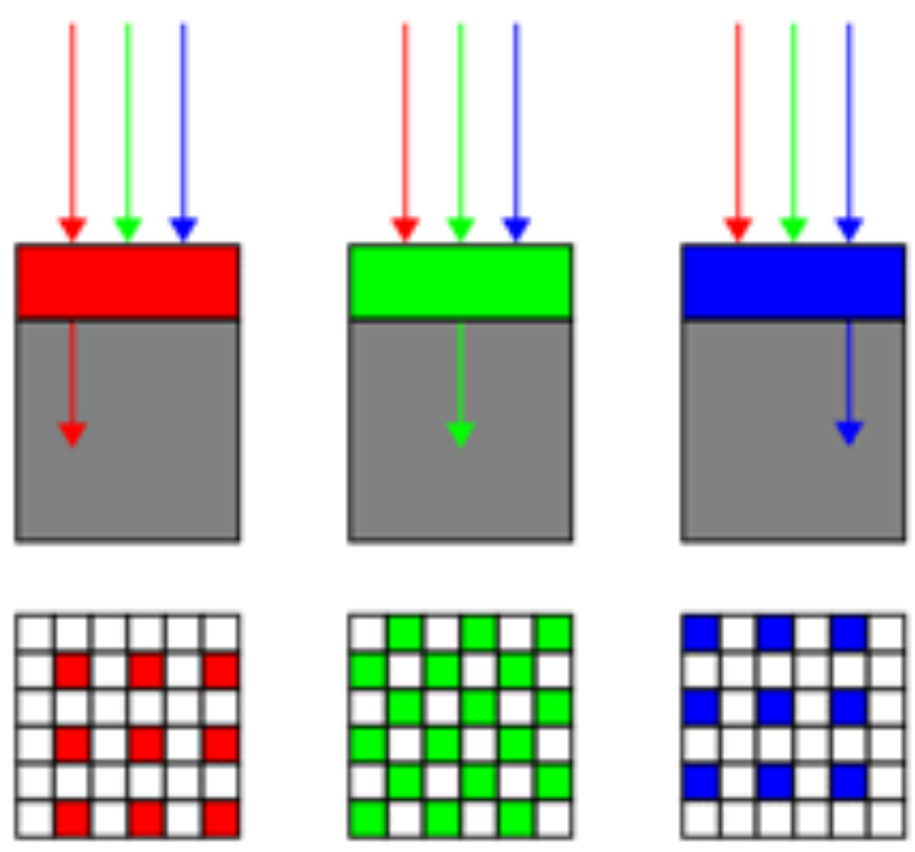
images as data

- an image can be represented in many ways
- most common format is a matrix of pixels
  - each “pixel” is BGRA



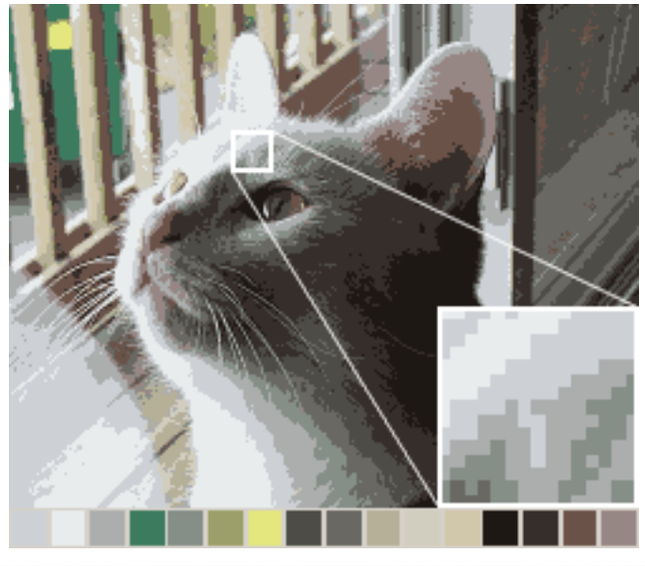
- used for capture and display





images as signals

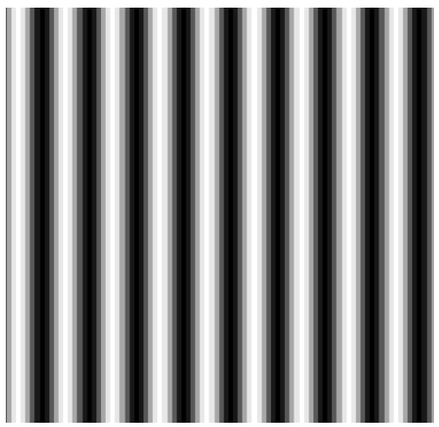
- everything from audio still applies
- quantization
  - each pixel can only take on 0-255 values
  - i.e., “stretching” in low light conditions



sampling errors

- in time (video)

- in space (resolution)
  - “frequency” is in terms of spatial sine waves



images as signals

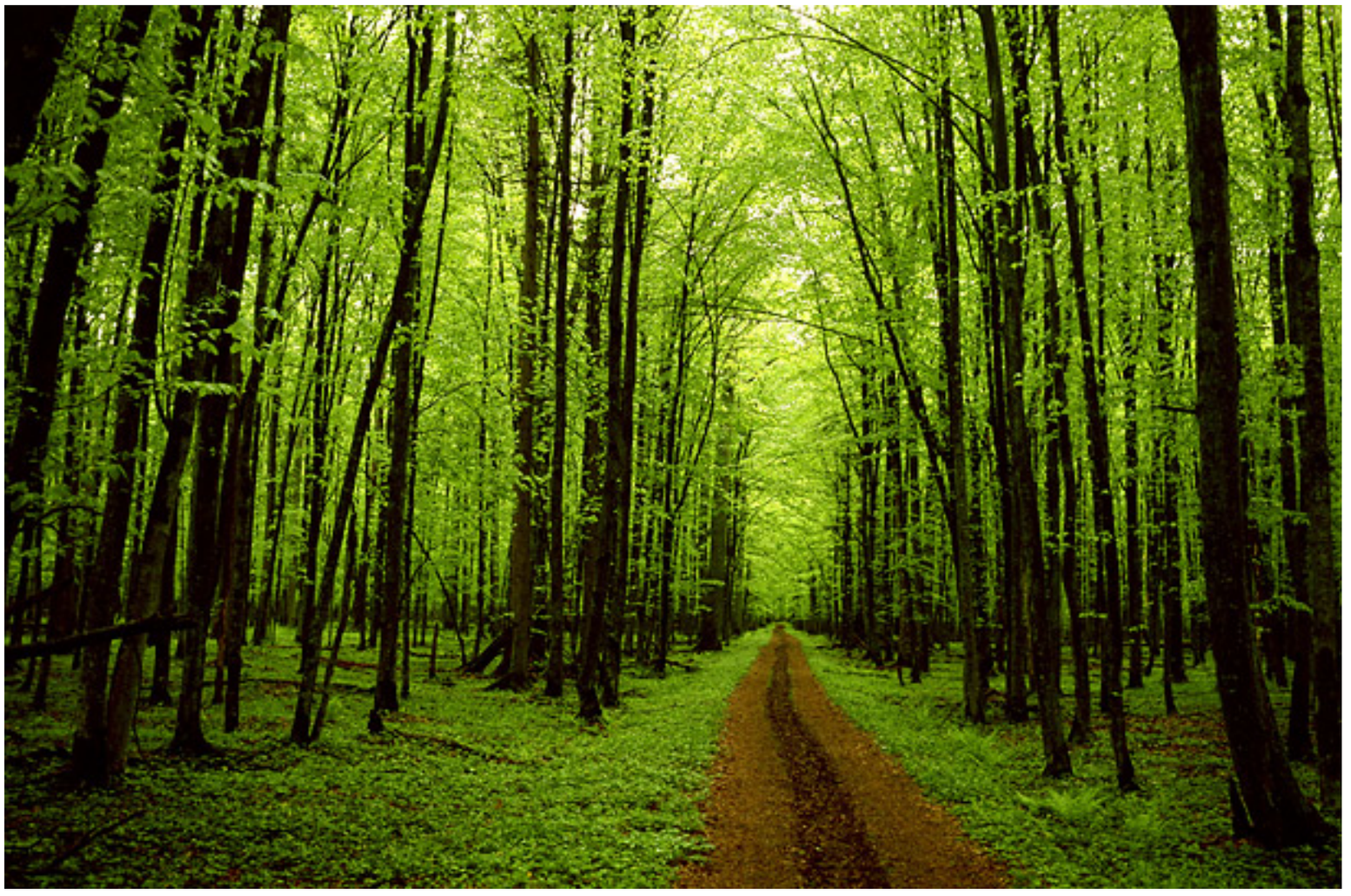


image with lots of high frequency

images as signals

image with lots of low frequency

what is filtering?

- same as audio
  - convolution (linear)

.11	.11	.11
.11	.11	.11
.11	.11	.11

kernel

averaging

image

1.1	1.5	2.4	2.7	4.3	3.2
1.6	2.3	4.0	4.7	6.8	4.8
1.8	2.6	4.5	5.6	7.5	5.1
1.4	2.2	4.3	6.1	8.0	5.2
1.4	2.3	4.5	6.3	8.0	5.2
1.3	2.1	4.3	5.8	7.7	5.1
1.2	1.7	3.3	4.1	5.2	3.3

what is filtering?

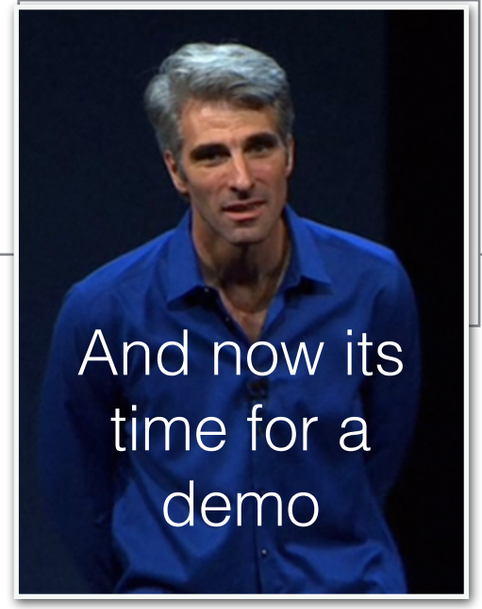
-1	0	1
-1	0	1
-1	0	1

vertical difference

-8	-2	-2	-7	-8	4
-12	-3	-6	-13	-7	5
-8	-2	-10	-15	-2	2
-8	-2	-16	-17	0	3
-8	-3	-17	-16	2	4
-8	-3	-16	-15	0	1
-8	-2	-9	-10	2	2

interesting operations

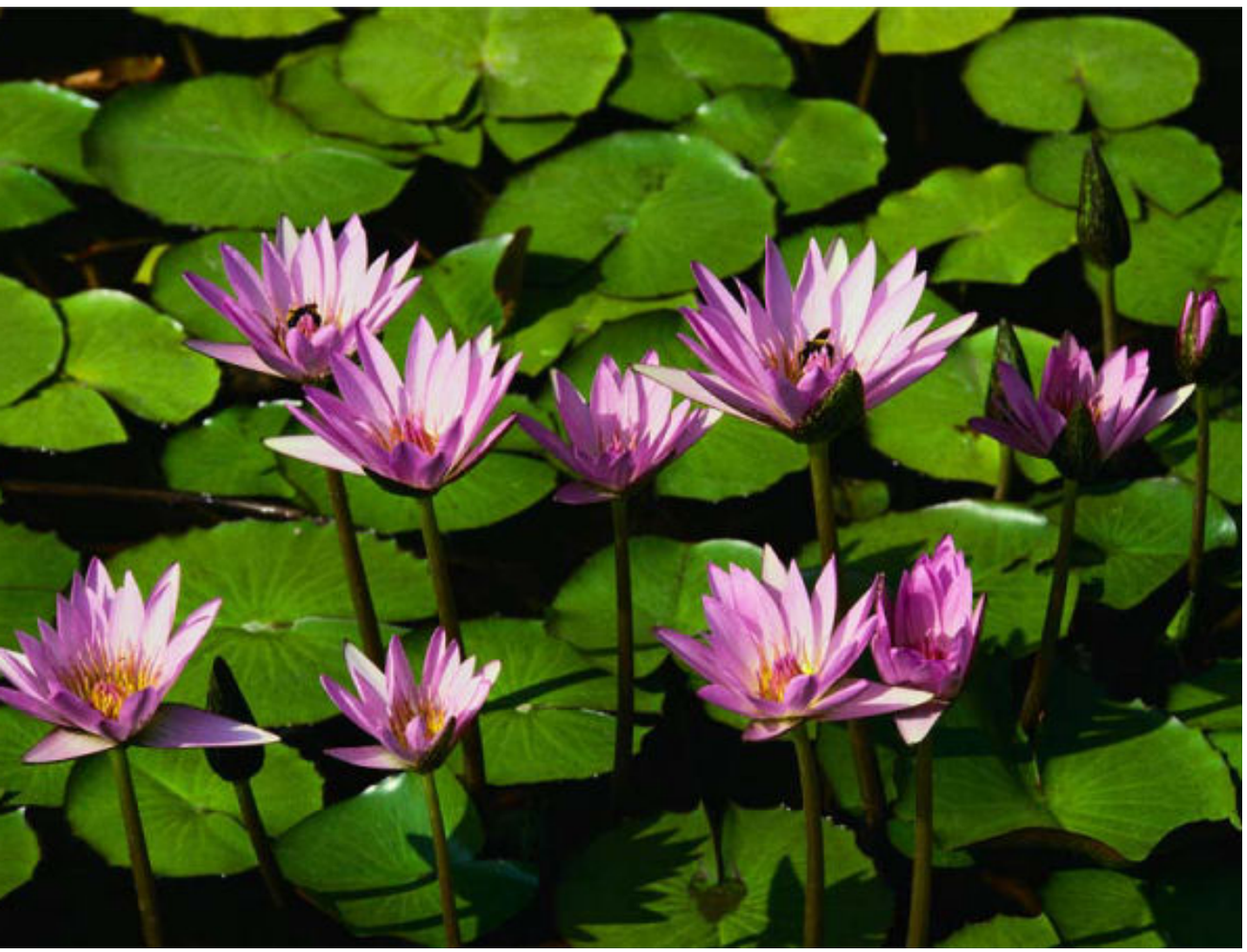
- the power of simple filtering
  - **a demo, via python**

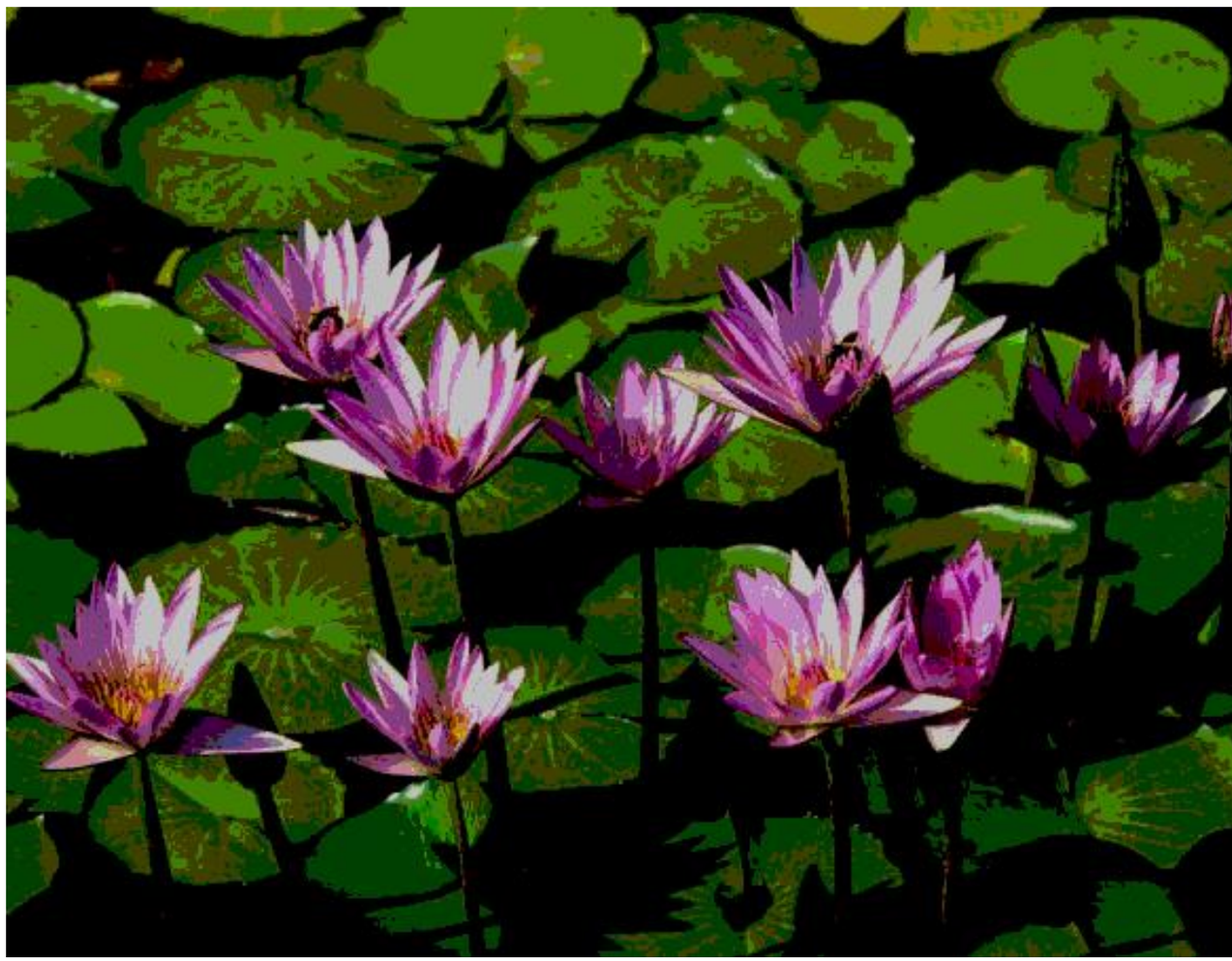


interesting operations

- the power of simple filtering
  - **a demo, via python**

but there is no need to just perform linear convolution!





non-linear filtering

-	-	-
-	max	-
-	-	-

kernel

-	-	-
-	median	-
-	-	-

gray scale image

4	4	5	6	9	9
4	4	8	8	9	9
4	4	8	8	8	8
4	4	9	9	9	9
4	4	7	9	9	9
4	4	9	9	9	9
4	4	9	9	9	9

filtering with color

-	-	-
-	max	-
-	-	-

R

G

B

1	4	2	5	6	9
1	4	2	5	5	9
1	4	2	8	8	7
3	4	3	9	9	8
1	0	2	7	7	9
1	4	3	9	8	6
2	4	2	8	7	9

filtering video

- for default 5S available in class (it gets better with newer phones):
- back camera is capable of capturing
  - 8MP photos (~30 MB raw)
  - 1080p HD video at 30 fps
- face camera
  - 1.2MP photos
  - 720p HD video at 30 fps

- video on the face camera is 1280x720 x3 channels x30fps
  - 82.9 million samples per second

so much data !!!

- we need to hardware accelerate
- look back to audio:
  - why is this:

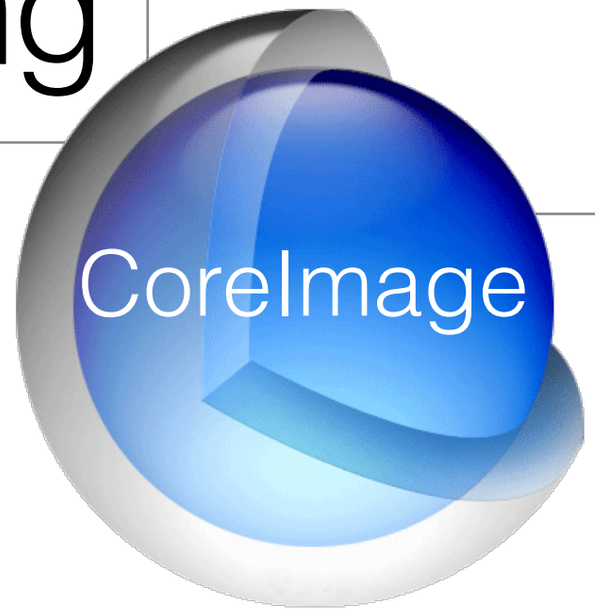
```
float one = 1.0;
vDSP_vdbcon(fftMagnitudeBuffer,1,&one,fftMagnitudeBuffer,1,kBufferLength/2,0);
```

  - faster than this:

```
for(int i=0;i<kBufferLength/2;i++){
        fftMagnitudeBuffer[i] = 20*logb(fftMagnitudeBuffer[i]);
    }
```

parallelized data processing images: GPU

options for image processing



- **CoreImage** (written by Apple)
  - somewhat extensible, very fast, easy to use
  - if not implemented, you can't do it (like computer vision), always getting better, can fall back on CPU
- **GPUImage** (independent developer)
  - open source, updated for swift, lots of users, very fast (comparable to CI), very easy to use
  - developed and maintained by one guy, Brad Larson
- **OpenCV** (started by Intel)
  - slow, requires c++, bloated
  - most comprehensive functionality, biggest user base (gigantic),

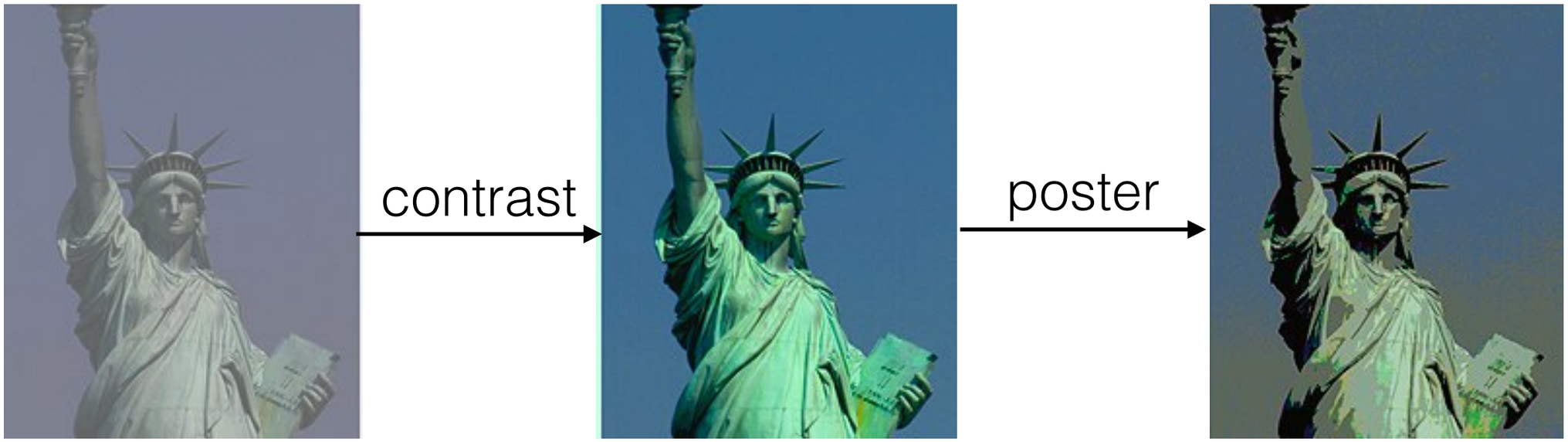
core image framework

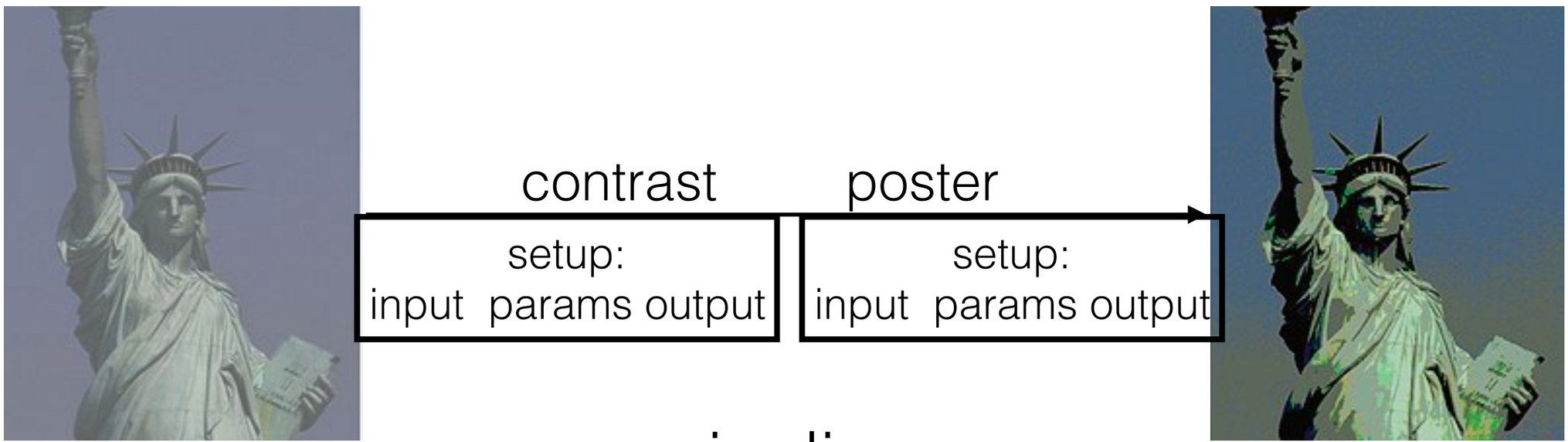
- defines images as CIImage instances
- defines a set of filters that (can be) GPU accelerated
  - optimizes filters when cascaded
- filters created through CIFilter class instances

CIAdditionCompositing	CIColorCrossPolynomial	CIFourfoldReflectedTile	CIMaximumComponent	CISourceAtopCompositing
CIAffineClamp	CIColorPolynomial	CIFourfoldRotatedTile	CIMaximumCompositing	CILinearToSRGBToneCurve
CIAffineTile	CIColorPosterize	CIFourfoldTranslatedTile	CIMinimumComponent	CISRGBToneCurveToLinear
CIAffineTransform	CIConstantColorGenerator	CIGammaAdjust	CIMinimumCompositing	CISourceInCompositing
CIBarsSwipeTransition	CIConvolution3X3	CIGaussianBlur	CIModTransition	CISourceOutCompositing
CIBlendWithMask	CIConvolution5X5	CIGaussianGradient	CIMultiplyBlendMode	CISourceOverCompositing
CIBloom	CIConvolution9Horizontal	CIGlideReflectedTile	CIMultiplyCompositing	CIStarShineGenerator
CIBumpDistortion	CIConvolution9Vertical	CIGloom	CIOverlayBlendMode	CIStraightenFilter
CICheckerboardGenerator	CICopyMachineTransition	CIHardLightBlendMode	CIPerspectiveTile	CIStripesGenerator
CICircleSplashDistortion	CICrop	CIHatchedScreen	CIPerspectiveTransform	CISwipeTransition
CICircularScreen	CIDarkenBlendMode	CIHighlightShadowAdjust	CIPinchDistortion	CITemperatureAndTint
CIColorBlendMode	CIDifferenceBlendMode	CIHoleDistortion	CIPixellate	CIToneCurve
CIColorBurnBlendMode	CIDisintegrateWithMask	CIHueAdjust	CIRadialGradient	CITriangleKaleidoscope
CIColorControls	CIDissolveTransition	CIHueBlendMode	CIRandomGenerator	CITwelvefoldReflectedTile
CIColorCube	CIDotScreen	CILanczosScaleTransform	CISaturationBlendMode	CITwirlDistortion
CIColorDodgeBlendMode	CIEightfoldReflectedTile	CILightenBlendMode	CIScreenBlendMode	CIUnsharpMask
CIColorInvert	CIExclusionBlendMode	CILightTunnel	CISepiaTone	CIVibrance
CIColorMap	CIExposureAdjust	CILinearGradient	CISharpenLuminance	CIVignette
CIColorMatrix	CIFaceDetector	CILineScreen	CISixfoldReflectedTile	CIVortexDistortion
CIColorMonochrome	CIFalseColor	CILuminosityBlendMode	CISixfoldRotatedTile	CIWhitePointAdjust
CIColorClamp	CIFlashTransition	CIMaskToAlpha	CISoftLightBlendMode	CIQRCodeGenerator

core image framework

- nothing happens until the image is rendered!





core image syntax

- Loading an image from the bundle
  - we need a CIImage instance, which stores more than just pixels (**note**: not a UIImage!)

```
let urlPath = NSBundle.mainBundle().pathForResource("smu-campus", ofType: "jpg")
let fileURL = NSURL.fileURLWithPath(urlPath!)

let beginImage = CIImage(contentsOfURL: fileURL)

        …processing here…

self.imageView.image = UIImage(CIImage: beginImage)
```

get image path from bundle

load image

show inside a UIImageView

core image syntax

```
let urlPath = NSBundle.mainBundle().pathForResource("smu-campus", ofType: "jpg")
let fileURL = NSURL.fileURLWithPath(urlPath!)

let beginImage = CIImage(contentsOfURL: fileURL)

let filter = CIFilter(name: "CIBloom")!
filter.setValue(beginImage, forKey: kCIInputImageKey)
filter.setValue(0.5, forKey: kCIInputIntensityKey)

outputImage = filter.outputImage!

self.sourceImageView.image = UIImage(CIImage: outputImage)

self.imageView.image = UIImage(CIImage: beginImage)
```

create filter

filter type

set parameters

input image

thresholds

get output

processing

core image filters

- https://developer.apple.com/library/mac/documentation/graphicsimaging/reference/CoreImageFilterReference/Reference/reference.html
- names, parameters, examples, etc.

Filters (linear and non-linear)
- CICategoryBlur
- CICategorySharpen
- CICategoryStylize

Distortion
- CICategoryCompositeOperation
- CICategoryDistortionEffect
- CICategoryGeometryAdjustment
- CICategoryTileEffect

Other Images
- CICategoryGenerator
- CICategoryHalftoneEffect
- CICategoryGradient

Color
- CICategoryColorAdjustment
- CICategoryColorEffect

Features (histogram, averages)
- CICategoryReduction

core image filters

CIBloom

Softens edges and applies a pleasant glow to an image.

Parameters

inputImage

A `CIImage` object whose display name is Image.

inputRadius

An `NSNumber` object whose attribute type is `CIAttributeTypeDistance` and whose display name is Radius.

Default value: 10.00

inputIntensity

An `NSNumber` object whose attribute type is `CIAttributeTypeScalar` and whose display name is Intensity.

Default value: 1.00

Member of

`CICategoryBuiltIn`, `CICategoryStillImage`, `CICategoryVideo`, `CICategoryStylize`

Localized Display Name

Bloom

Figure 8 The result of using the CIBloom filter

Availability

Available in OS X v10.4 and later and in iOS 6.0 and later.

```
radius = 100;
CIFilter *filter =
[CIFilter filterWithName:@"CIBloom" keysAndValues:
          @"inputImage", myImage,
          @“inputRadius", @(radius),
          @"inputIntensity", @0.5,
          nil];
```

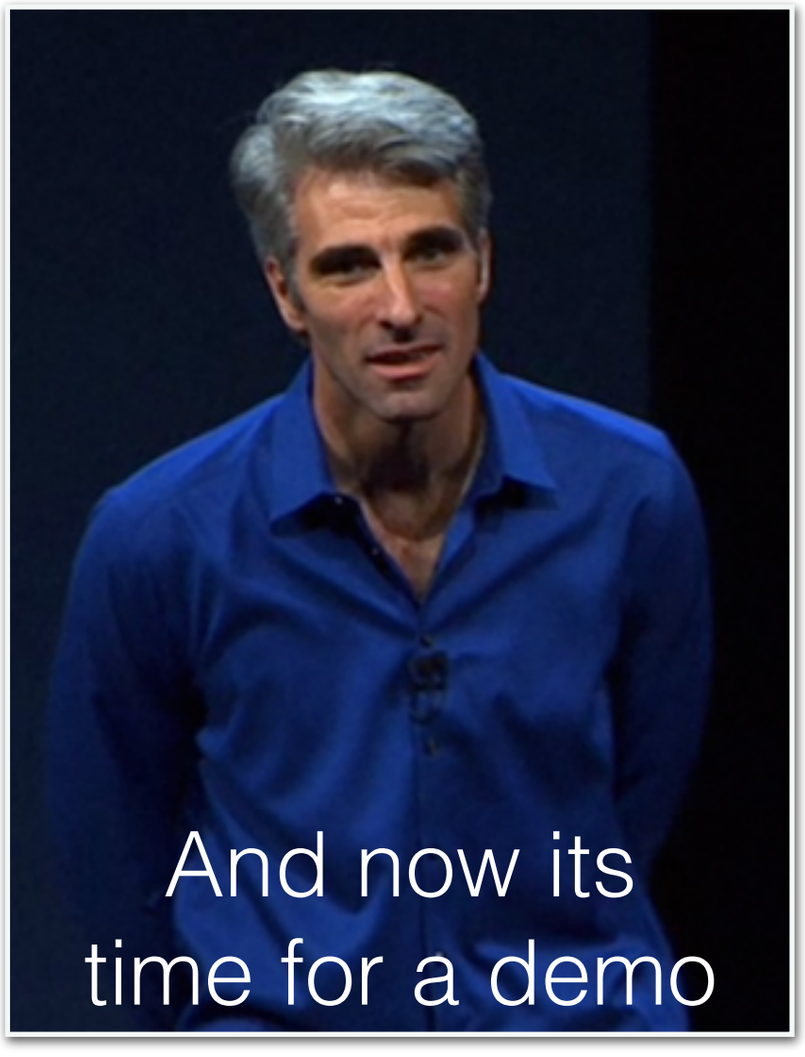
```
CIFilter *filter =
[CIFilter filterWithName:@“CIBloom”];

[filter setValue:myImage
          forKey:kCIInputImageKey];
```

available?

core image demo

- ImageLab, filter image from bundle

http://www.raywenderlich.com/76285/beginning-core-image-swift



custom filters

```
const CGFloat weights[] = { 1, 0, -1,
                            2, 0, -2,
                            1, 0, -1};
```

```
result =
[CIFilter filterWithName:@"CIConvolution3X3" keysAndValues:
                         @"inputImage", inputImage,
                         @"inputWeights",
                                [CIVector vectorWithValues:weights count:9],
                         @"inputBias", @0.5,
                    nil].outputImage;
```

chaining filters

```
NSMutableArray *filters = [[NSMutableArray alloc]init];
  [filters addObject:[CIFilter filterWithName:@"CISepiaTone"]];
  [filters addObject:[CIFilter filterWithName:@"CIBloom"]];
  [filters addObject:[CIFilter filterWithName:@"CIColorInvert"]];

  outputImage = inputImage;
   for(CIFilter *filter in filters){
       [filter setValue:outputImage forKey:kCIInputImageKey];
       outputImage = filter.outputImage;
   }
```

beyond the main bundle

- easy to get photos!
- demos in the branch of **ImageLab**, if you are interested!
- from library:
  - use the `UIImagePickerControllerDelegate` protocol
- from camera:
  - use the `UIImagePickerControllerDelegate` protocol
  - `cameraUI.sourceType = UIImagePickerControllerSourceTypeCamera`
- https://developer.apple.com/library/ios/documentation/AudioVideo/Conceptual/CameraAndPhotoLib_TopicsForIOS/Introduction/Introduction.html#//apple_ref/doc/uid/TP40010405-SW1

access still images in album

```
<UIImagePickerControllerDelegate, UINavigationControllerDelegate>


- (IBAction)pickImageFromAlbum:(id)sender {
    UIImagePickerController *myPicker =
            [[UIImagePickerController alloc] init];
    myPicker.delegate = self;
   [self presentViewController:myPicker animated:YES completion:nil];
}

- (void)imagePickerControllerDidCancel:
(UIImagePickerController *)picker {
    [self dismissViewControllerAnimated:YES completion:nil];
}


- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info
{
    [self dismissViewControllerAnimated:YES completion:nil];
    UIImage *pickedImage =
            [info objectForKey:UIImagePickerControllerOriginalImage];
}
```

this example uses a button

allocate & set options

present modal view

interact with modal VC

get image from VC

take a still image photo

```
<UIImagePickerControllerDelegate, UINavigationControllerDelegate>

- (IBAction)pickImageFromAlbum:(id)sender {
    UIImagePickerController *myPicker =
            [[UIImagePickerController alloc] init];
    myPicker.delegate = self;

    myPicker.sourceType = UIImagePickerControllerSourceTypeCamera;

   [self presentViewController:myPicker animated:YES completion:nil];
}
```

```
- (void)imagePickerControllerDidCancel:
(UIImagePickerController *)picker {
    [self dismissViewControllerAnimated:YES completion:nil];
}


- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info
{
    [self dismissViewControllerAnimated:YES completion:nil];
    UIImage *pickedImage =
            [info objectForKey:UIImagePickerControllerOriginalImage];
}
```

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 screen
- 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

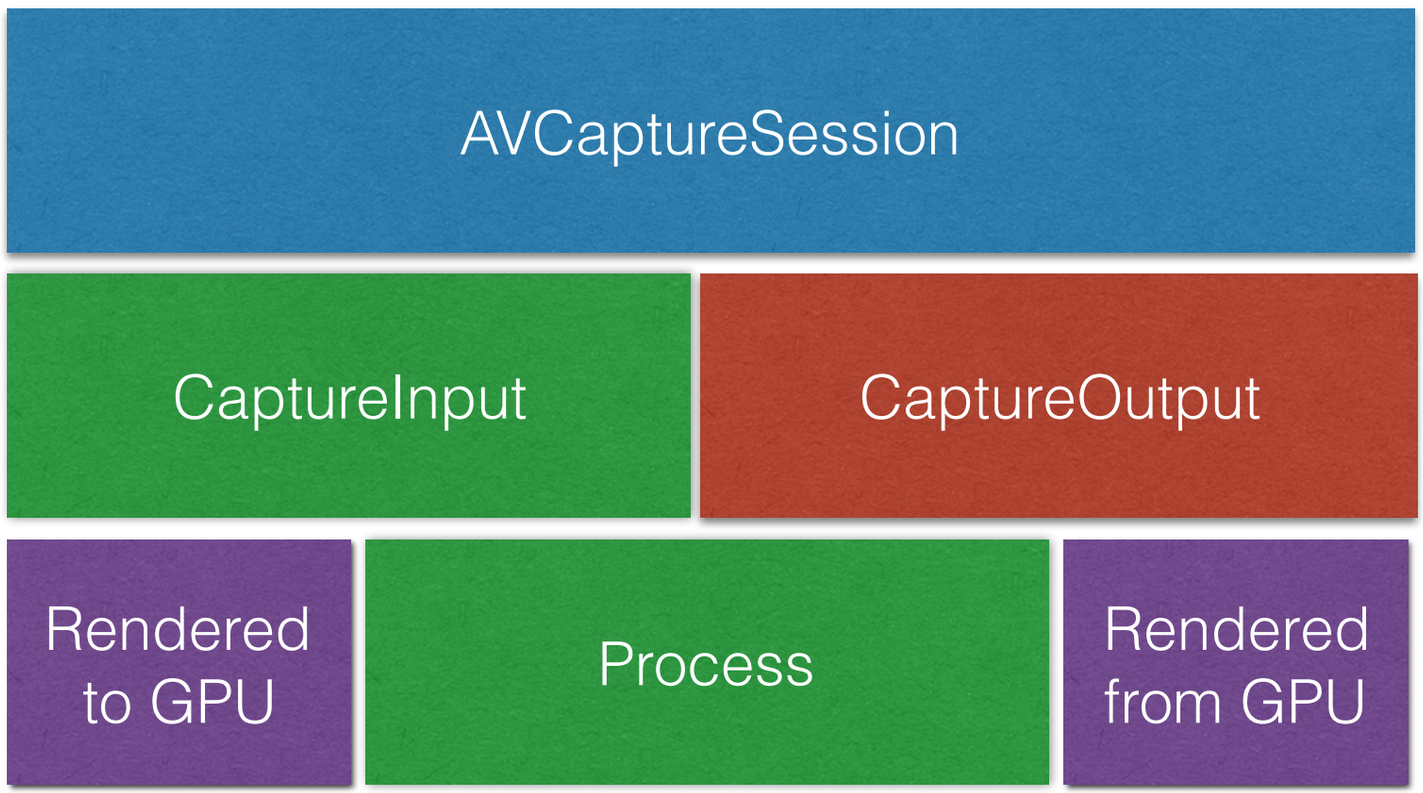
- setup the capture
  - device: front or back camera

```
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!

how to program this?

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

declare

```
#import "VideoAnalgesic.h"

@property (strong,nonatomic) VideoAnalgesic *videoAnalgesic;
```

init

```
self.captureManager = [VideoAnalgesic captureManager];
```

start

```
if(![self.captureManager isRunning])
     [self.captureManager start];
```

options

```
self.captureManager.preset = AVCaptureSessionPresetMedium;
[self.captureManager setCameraPosition:AVCaptureDevicePositionFront];
```

stop

```
[self.captureManager stop];
```

VideoAnalgesic

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

```
[self.captureManager setProcessBlock:^(CIImage *cameraImage){
    return cameraImage;
}];
```

return image to draw to screen

image from camera sent in block

```
__block CIFilter *filter = [CIFilter filterWithName:@"CISepiaTone"];
[self.captureManager setProcessBlock:^(CIImage *cameraImage){

    [filter setValue:cameraImage forKey:kCIInputImageKey];
    CIImage* output = filter.outputImage;
    return output;

}];
```

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.sharedInstance
```

start

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

options

```
self.videoAnalgesic.setPreset(AVCaptureSession.Preset.medium)
self.videoAnalgesic.toggleCameraPosition()
self.videoAnalgesic.setCameraPosition(AVCaptureDevice.Position.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
         }
```

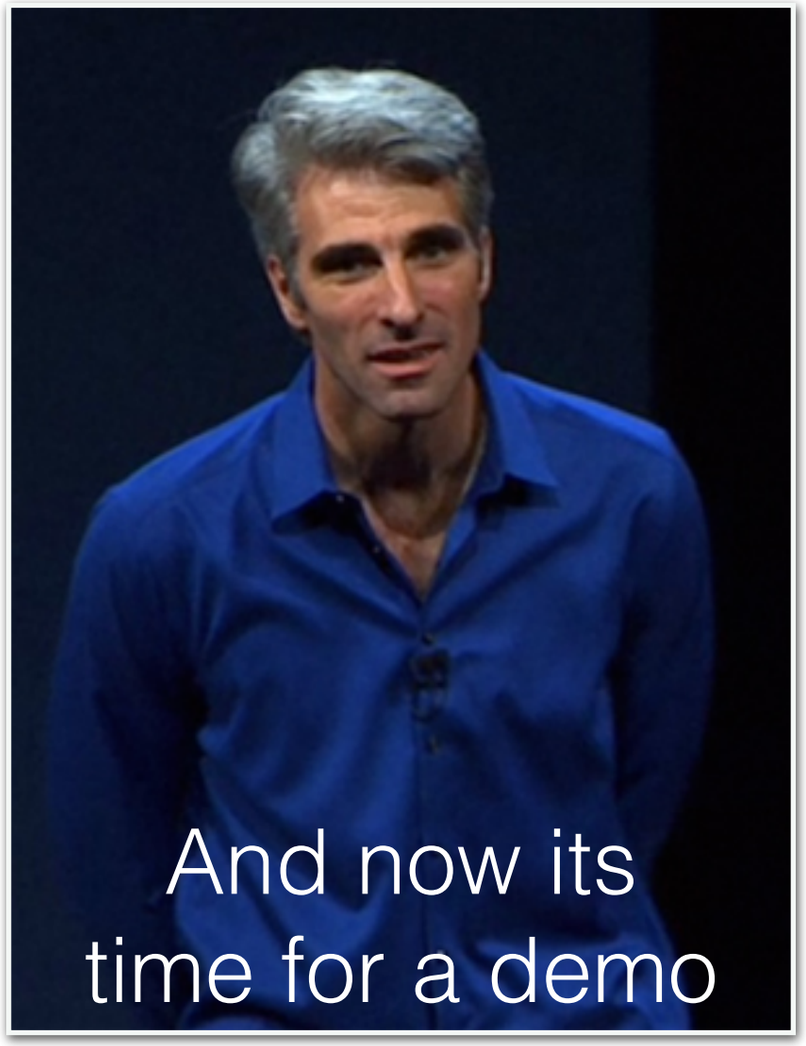
image from camera passed in

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++

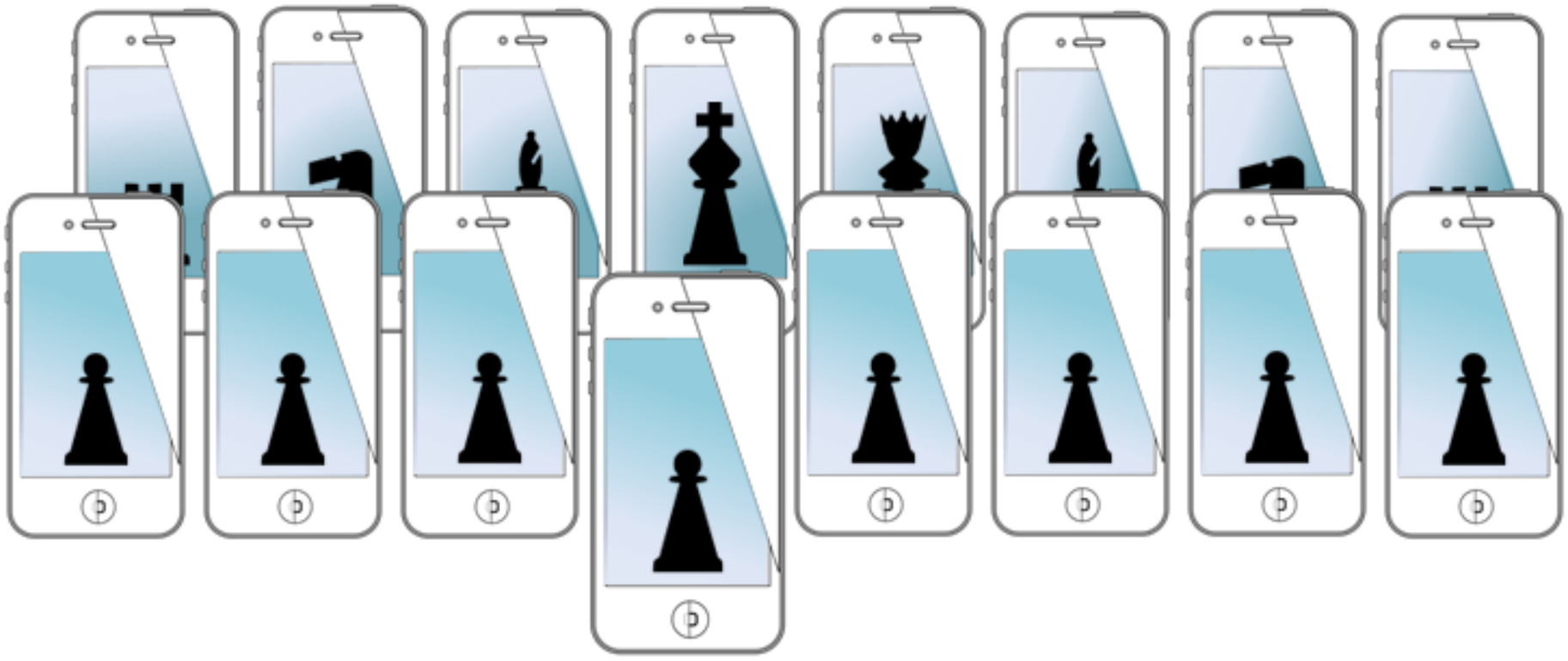


for next time…

- filters
- faces
- physiology

- then on to the next in-class assignment!!
  - computer vision with OpenCV
    - generic operations
    - tracking

MOBILE SENSING LEARNING



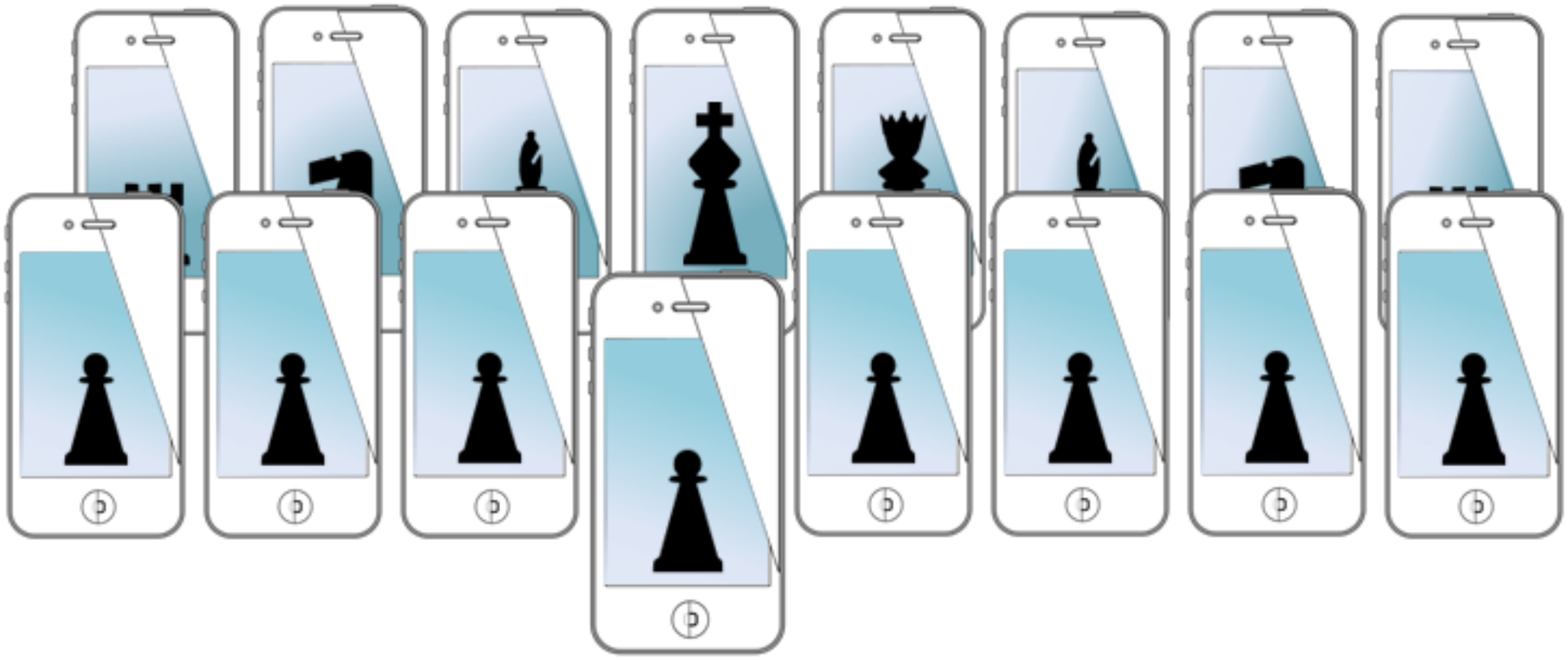
CSE5323 & 7323

Mobile Sensing and Learning

week 6: core image and image processing

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

MOBILE SENSING LEARNING



CSE5323 & 7323

Mobile Sensing and Learning

week 6: computer vision with core image

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

course logistics

- Grades are coming
- A3 is due Friday!
- A4 is due 2 weeks from Friday
- A4 constraints on website
- **next lecture: in-class assignment, OpenCV**

agenda

- video processing
- computer vision
  - face detection
  - heart physiology

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

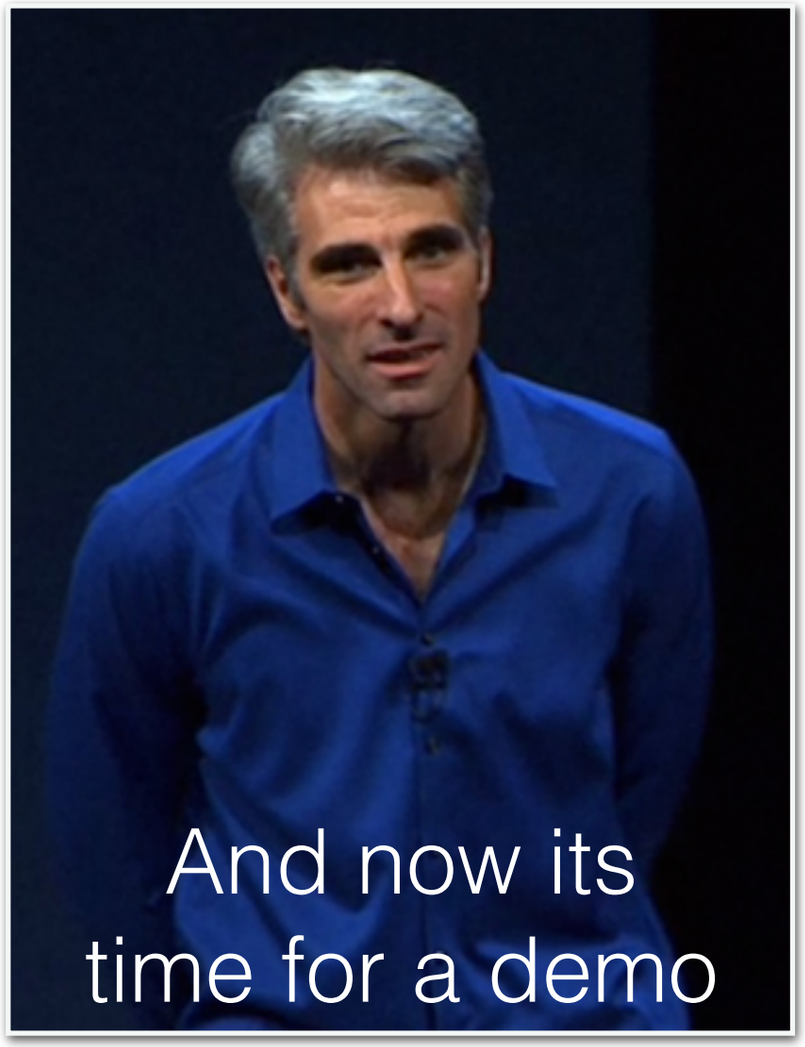
Transform from UI to CoreImage

```
@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

filter param demo

- PinchMe



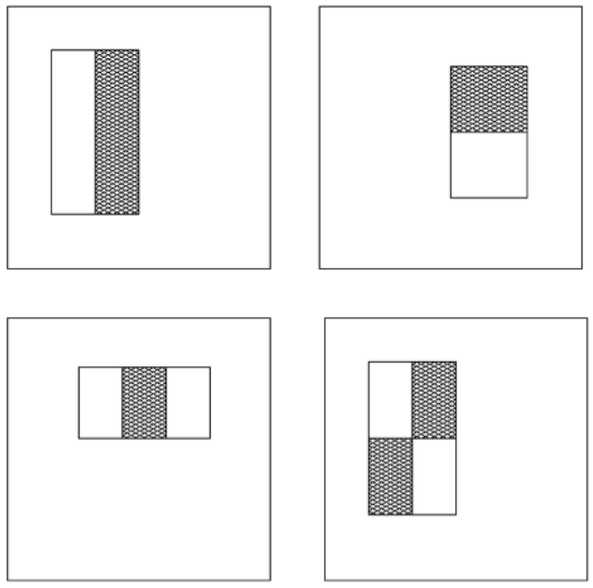
face detection

- is a face in the picture and where?
- algorithm is probably hardware accelerated variant of Viola Jones
- essentially, a “matching” filter is applied
  - only happens in one orientation
  - but multiple scales (which takes “some” time)



an intuition

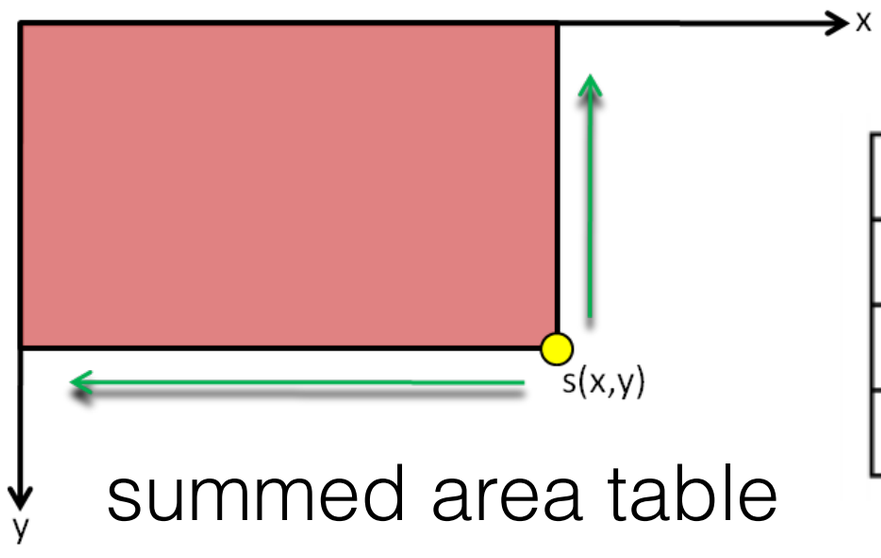
- face detection with “rectangle” features



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



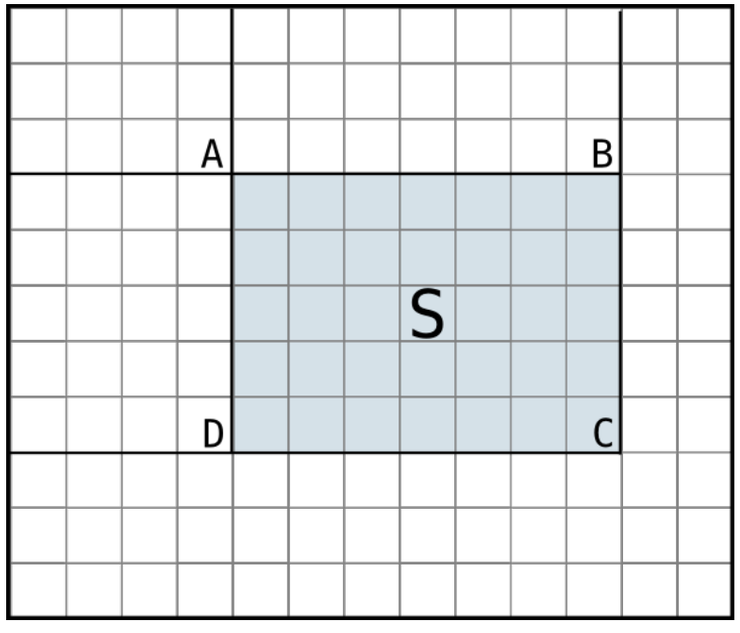
summed area table

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

original

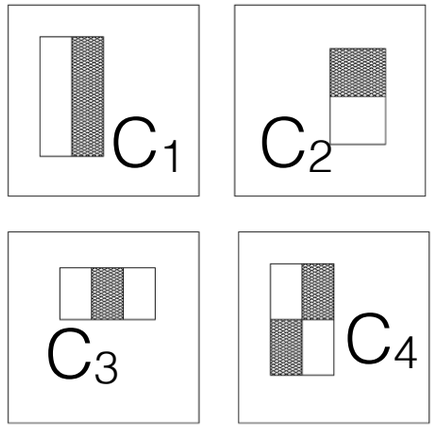
4	5	7	9
4	9	12	17
7	13	16	25
9	16	22	33

summed



learning

- train a bunch of “classifiers” with lots of examples



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

combine output of classifiers

cascade these

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

learning

- 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 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

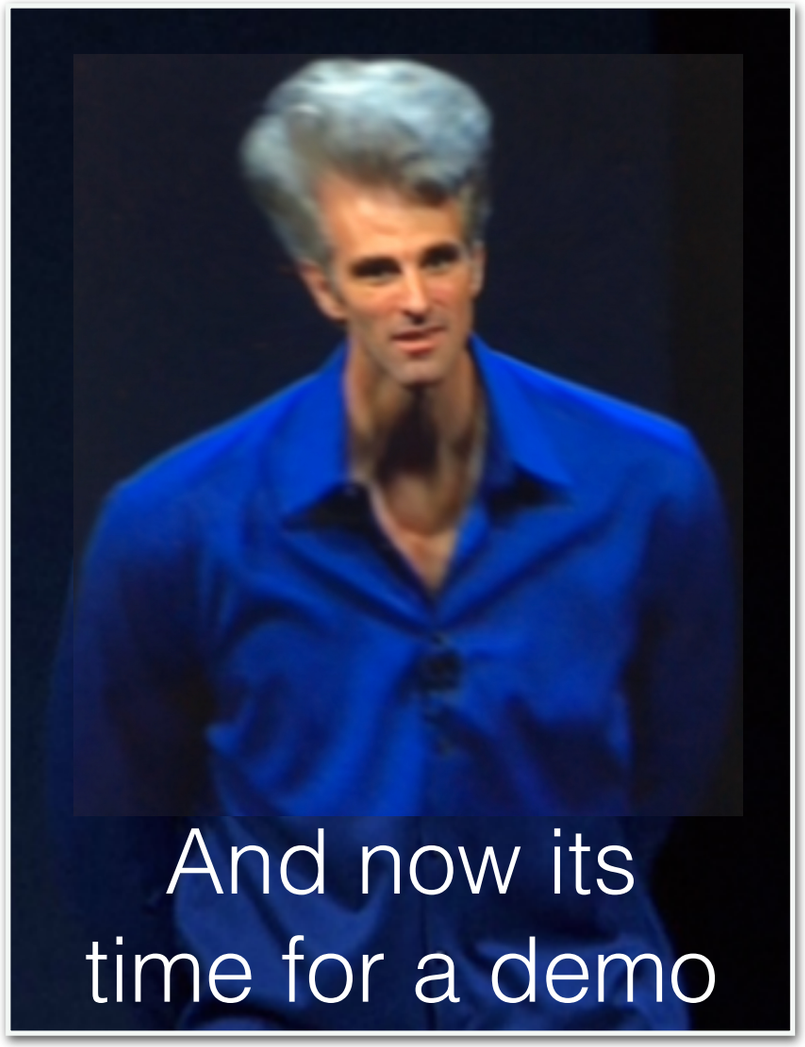
```
var features = detector.featuresInImage(inputImage, options: optsFace)

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

do this

options specific to "run"

face demonstration

- PinchMe++



face detection

- many tracking mechanisms are supported
- eye location
- mouth location
- smile detection
- blink / wink detection for each eye
- all use a variant of the Haar Wavelet method (probably)

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

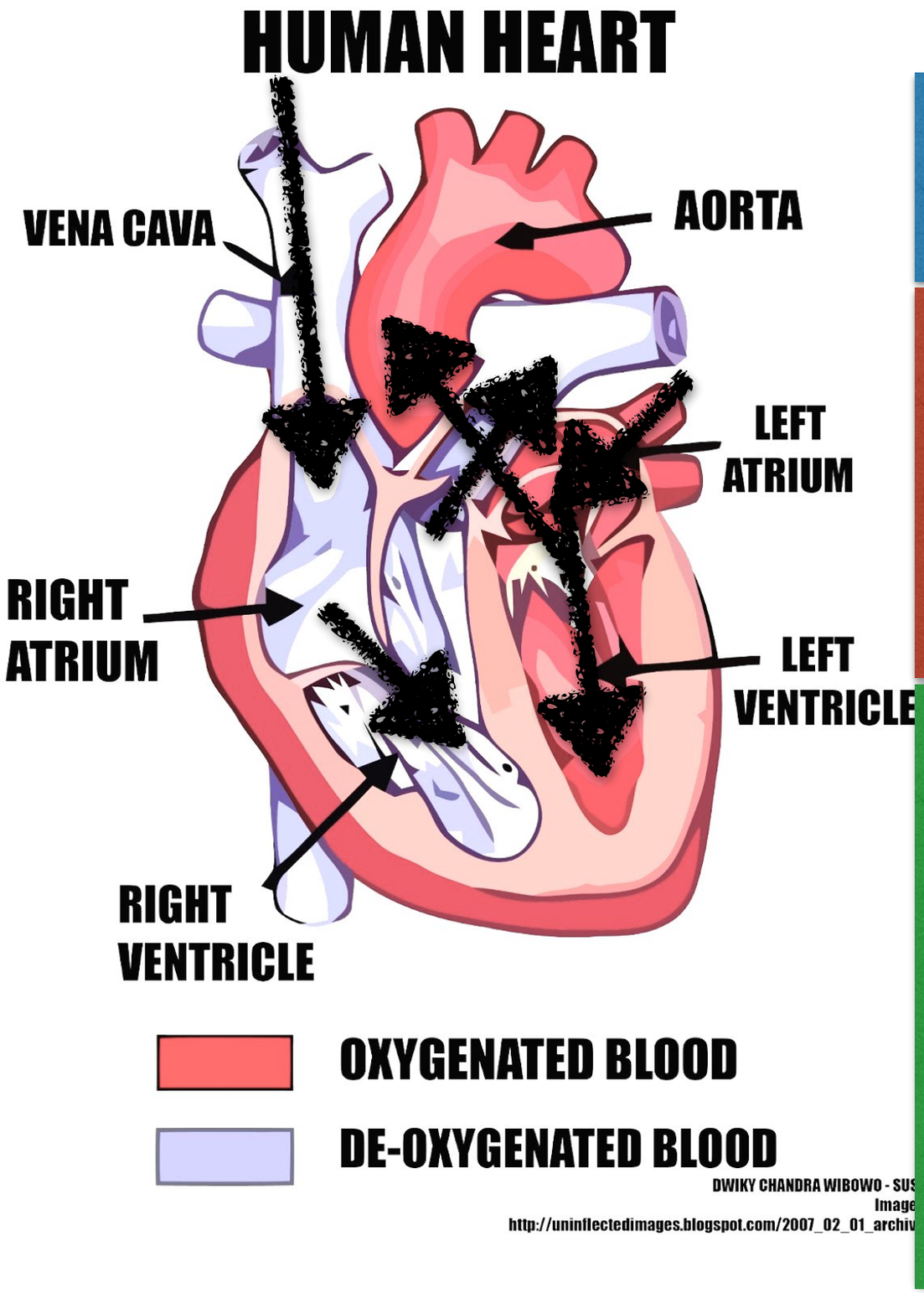
computer vision in iOS

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

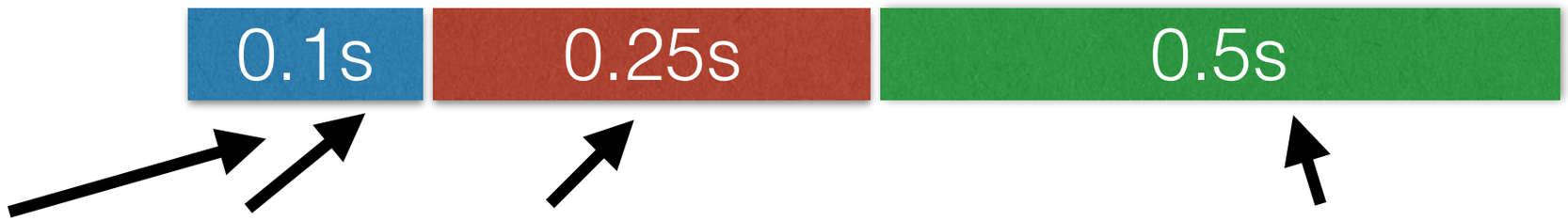
health?

- 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**

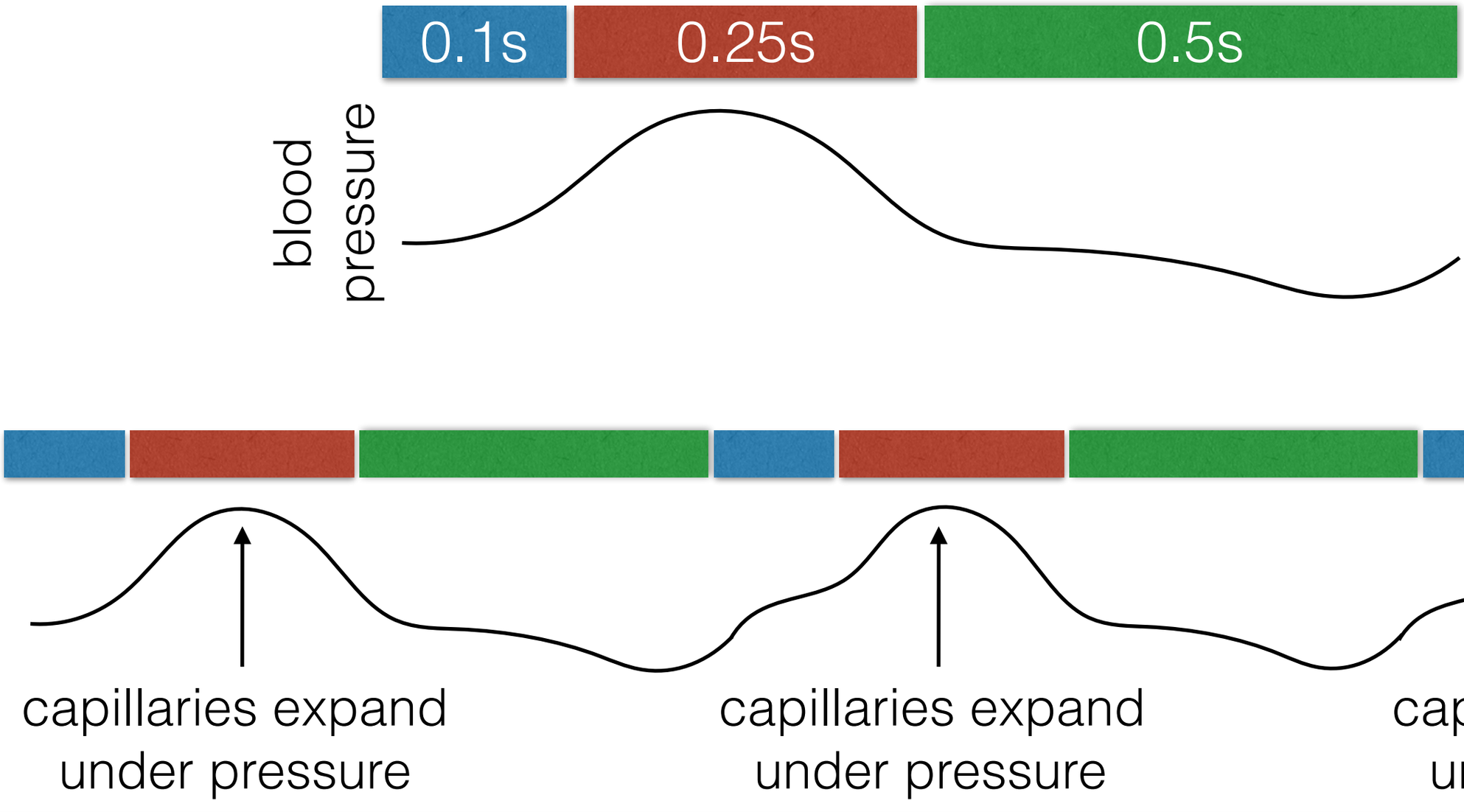
the cardiac cycle



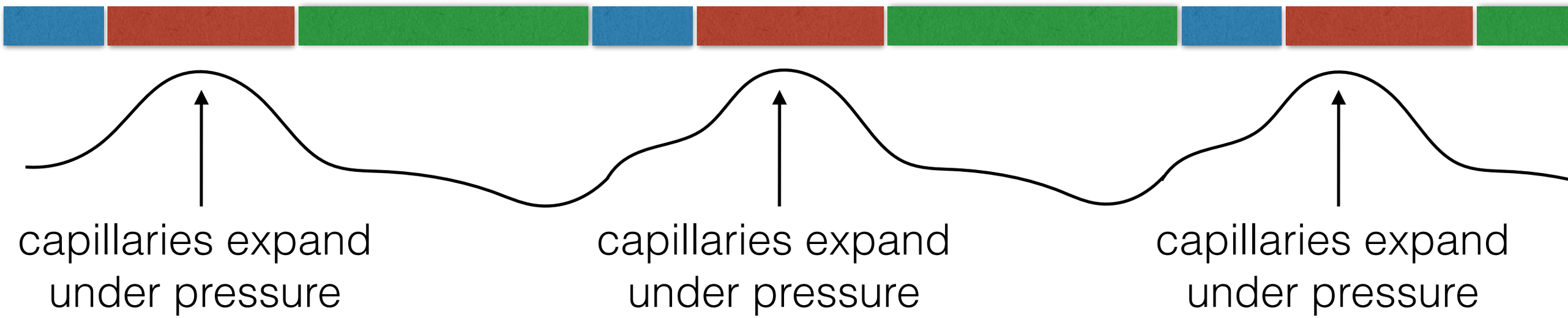
- aortic valve and vena cava valve open
- blood enters ventricles
- pump of heart ejects blood into arteries and out to lungs/body
- heart relaxes, letting blood flow into atria and ventricles


0.1s
0.25s
0.5s

a signal from the heart



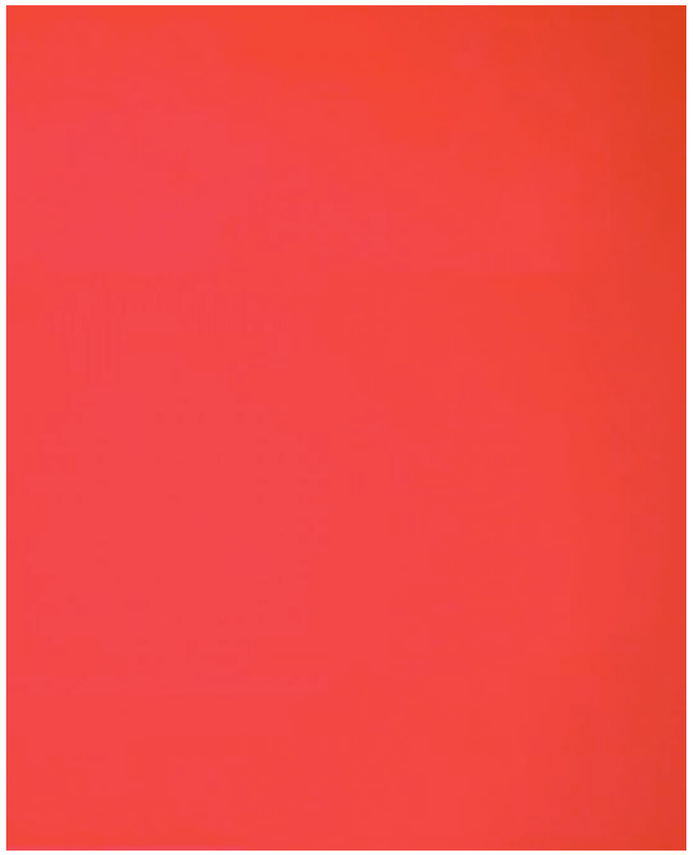
a signal from the heart



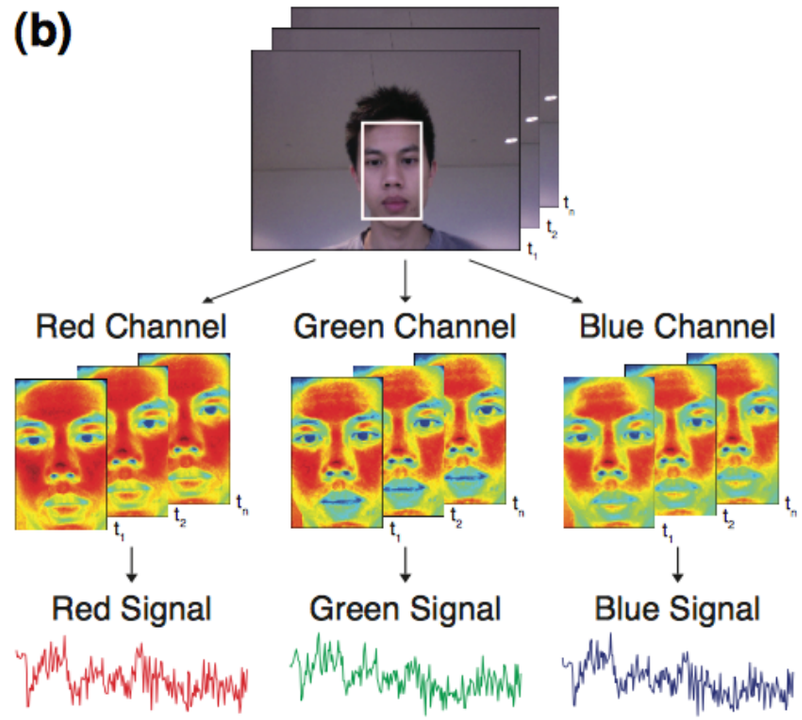
- capillary expansion means more blood under skin
  - shift in redness from oxygenated blood
  - shift in blueness from deoxygenated blood
  - more blood molecules for light to reflect from

a signal from the heart

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



exemplary work



photoplethysmography (PPG)

caveats

- do not press too hard on camera
- 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***

another cool example

digiDoc Technologies

Home Products

The World's Only Digital Solution

The Pulse Oximeter app measures both Heart Rate and Oxygen Saturation. The app integrates with Apple Health. There's no need for an external device. *Your iPhone is all you need.*

INTENDED USE

The Pulse Oximeter app is for use by sports users who are interested in knowing their blood oxygenation level (SpO2) and Heart Rate. The Pulse Oximeter app is NOT INTENDED FOR MEDICAL USE. The Pulse Oximeter app can be used in a wide range of settings, including between exercises, 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

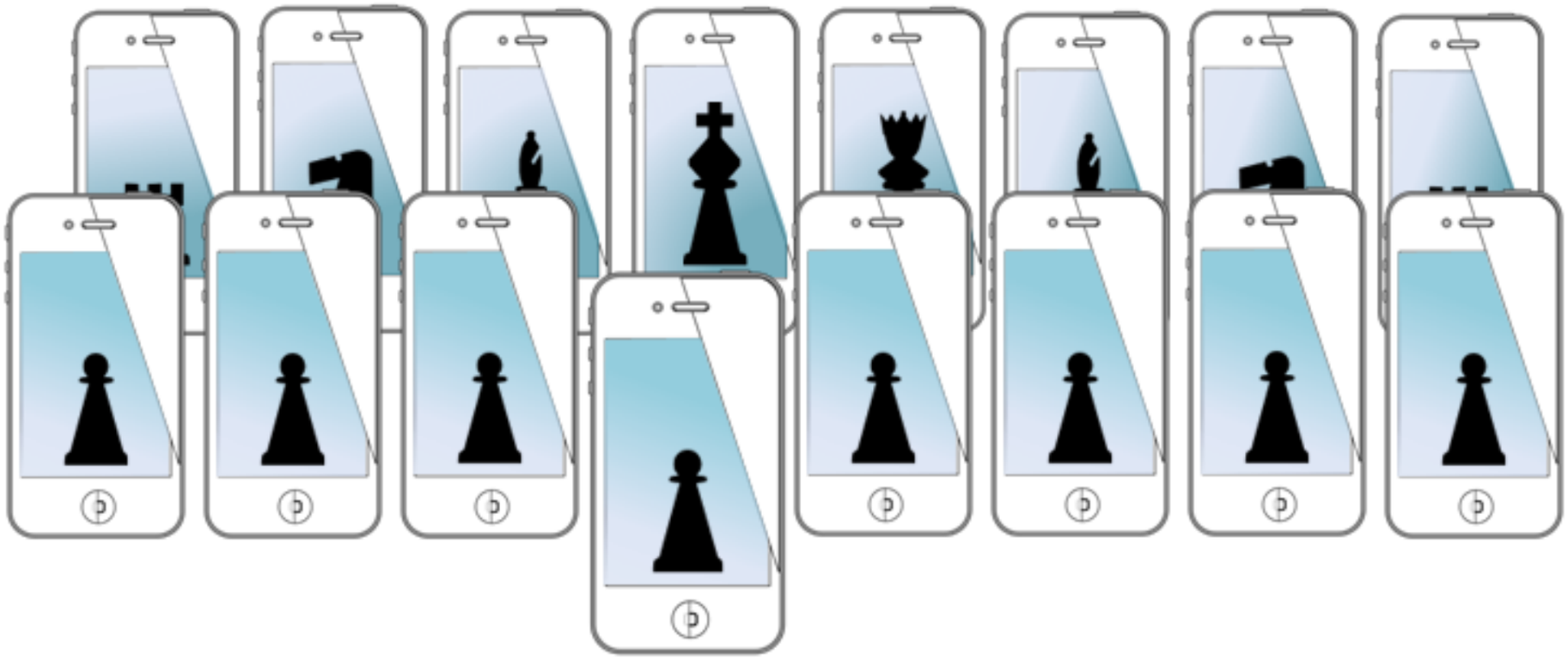
- Measure pulse and blood oxygen saturation
- Pulse Oximeter range 93-100%
- Record and store history of data
- Real-time PPG graph for immediate accuracy
- Apple HealthKit integration
- Label selection

Download on the App Store

for next time…

- computer vision with OpenCV
  - watch video lecture for OpenCV
  - fun operations in imaging
  - come ready to use OpenCV next time

MOBILE SENSING LEARNING



CSE5323 & 7323

Mobile Sensing and Learning

week 6: computer vision with core image

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